Primalidade em Tempo Polinomial

Uma introdução ao algoritmo AKS

S. C. Coutinho

Universidade Federal do Rio de Janeiro

2004

©S. C. Coutinho 2004

Para a minha mãe

Prefácio

As décadas finais do século XX, e este início de século XXI, têm sido marcados por progressos notáveis na solução de problemas matemáticos importantes, alguns dos quais estavam em aberto há séculos. Isto é especialmente verdadeiro no caso da teoria de números, com a solução da Conjectura de Mordell por G. Faltings em 1984, e as demonstrações do Último Teorema de Fermat por A. Wiles, em 1995, e da Conjectura de Catalan por P. Mihăilescu, em 2002. Entretanto, a demonstração de cada um destes resultados envolve uma seção transversal expressiva da matemática contemporânea, de modo que não são acessíveis, senão ao especialista.

Isto explica o enorme impacto produzido na comunidade matemática pela descoberta de um algoritmo polinomial de primalidade por Agrawal, Kayal e Saxena. Eles não apenas resolveram, de maneira brilhante, um problema que está conosco há milênios, como a solução é tão simples que pode ser compreendida por um estudante de graduação em matemática ou ciência da computação. Na verdade, a maioria das técnicas matemáticas utilizadas no algoritmo, e em sua demonstração, já eram conhecidas desde o final do século XIX. Havia apenas duas exceções: a noção de custo de um algoritmo, que só foi formalizada na segunda metade do século XX, e um resultado sobre a distribuição de números primos em intervalos, publicada em 1985.

Não espanta, portanto, que a divulgação na web, em agosto de 2002, do artigo original de Agrawal, Kayal e Saxena tenha provocado um *frisson* raramente visto, que se espalhou não apenas entre matemáticos e especialistas em computação, mas entre todos os interessados em matemática, ao ponto de virar notícia de jornal. É claro que a utlização dos primos nos sistemas de criptografia que garantem a segurança de nossas transações pela web deu um lustro prático ao que seria, em outros tempos, uma curiosidade matemática. Para completar a atmosfera meio mítica associada a este tipo de acontecimento, os autores da proeza eram três indianos, dois dos quais haviam acabado há pouco o curso de graduação.

Poucas semanas depois da divulgação do artigo na web, estava claro que o resultado estava correto, e começou a corrida para explorá-lo, melhorá-lo e generalizá-lo. É neste estágio que estamos atualmente, e ainda é prematuro dizer a que novos algoritmos as idéias introduzidas por Agrawal, Kayal e Saxena poderão conduzir. Contudo, dois anos depois de descoberto, o algoritmo já

foi simplificado a um ponto que é possível descrevê-lo de modo inteiramente elementar, e é exatamente este nosso objetivo.

Na verdade, o algoritmo exposto no capítulo 5 deste livro é uma variação, devida a H. W. Lenstra Jr do algoritmo que apareceu em **[Agrawal, Kayal e Saxena 2002]**. A versão que apresentamos segue a exposição de D. Bernstein **[Bernstein 2003]**, que por sua vez inclui simplificações sugeridas por Kiran Kedlaya. A missão do livro é tornar o algoritmo acessível, em todos os seus detalhes, a qualquer um que esteja familiarizado com o conteúdo de **[Coutinho 2001]**, ou algum livro equivalente. Isto significa que, além de explicar todo o procedimento do algoritmo, provamos que funciona corretamente e que tem custo polinomial. Como sempre acontece, foi necessário pagar um preço para poder manter o livro dentro destes limites. Assim, não foi possível formular o algoritmo da maneira mais eficiente possível, nem chegar a obter uma avaliação precisa de seu custo. Além disso, o livro se concentra na matemática que serve de suporte ao algoritmo, de modo que não abordamos em detalhe questões referentes à sua implementação. Mais informações sobre o que ficou de fora do livro, incluindo referências sobre a implementação do algoritmo, podem ser encontradas no epílogo.

Como já foi dito, o capítulo 5 deste livro está baseado essencialmente na exposição do AKS por D. Bernstein. A organização geral do material também deve muito a uma palestra sobre o assunto dada por R. Schoof no IMPA, em junho de 2003. A motivação para escrever o texto veio do interesse gerado pelo assunto entre os meus alunos da disciplina *Número Inteiros e Criptografia*, na Universidade Federal do Rio de Janeiro. Depois de tentar explicar o algoritmo em aula, menos de um mês depois de sua divulgação, ficou bastante claro que seria necessário entendê-lo com bastante detalhe, para poder responder às perguntas dos alunos. A partir daí passei a acompanhar, pela web, a evolução das idéias propostas no algoritmo. Logo ficou claro que uma exposição elementar dessas idéias seria, não apenas possível, como também desejável. A desculpa que faltava para pôr mãos à obra veio com o convite dos organizadores da XVIII Escola de Álgebra para dar um mini-curso sobre este assunto durante a escola.

Gostaria de agradecer a Hernando Bedoya e a Moysés Cohen pela leitura cuidadosa do livro, e por seus comentários, correções e sugestões, que melhoraram em muito o texto. Naturalmente, quaisquer erros que tenham restado são de minha inteira responsabilidade. Finalmente, agradeço à Sociedade Brasileira de Matemática por ter acolhido o texto em sua coleção de livros de iniciação científica.

Rio de Janeiro, 21 de maio de 2004

Conteúdo

Prefácio iii

Introdução 1

Capítulo 1. Primos 9
1. Primos 9
2. Custo de um algoritmo 12
3. Crivo de Eratóstenes 16
4. Distribuição dos primos 18
5. Exercícios 21

Capítulo 2. Grupos abelianos 25
1. Definição e exemplos 25
2. Subgrupos 28
3. Teorema de Lagrange 31
4. Grupos quocientes 34
5. Exercícios 38

Capítulo 3. Anéis, ideais e polinômios 41
1. Anéis 41
2. Anéis de polinômios 43
3. Divisão de polinômios 45
4. Ideais 49
5. Anéis quocientes 51
6. Fatoração única de polinômios 54
7. Exercícios 58

Capítulo 4. Testes de primalidade 61
1. Um teste determinístico de primalidade 61
2. Testes Não Determinísticos de Primalidade 65
3. Exponenciação 68
4. Balanço 70
5. Exercícios 71

Capítulo 5. O algoritmo AKS 75
1. O teorema fundamental 75
2. O algoritmo AKS 80

3. Custo do algoritmo 82
4. Exercícios 85

Epílogo 89

Apêndice 93
1. O método de Newton 93
2. O algoritmo 95
3. Exercícios 97

Bibliografia 99

Índice 103

Introdução

Até onde sabemos, o conceito de número primo foi introduzido na Grécia Antiga. Vamos encontrá-lo, por exemplo, nos *Elementos* de Euclides, escrito por volta de 300 a. C.. Segundo a definição 11 do Livro VII dos *Elementos*,

> Um número primo é aquele que é medido apenas pela unidade.

Na terminologia usada atualmente, a expressão 'medido por' traduz-se como 'divisível por um número menor do que ele próprio'. Assim, um número natural é *primo* se não é divisível por nenhum inteiro positivo menor que ele, exceto 1. Um pouco adiante, temos a definição de número composto:

> Um número composto é aquele que é medido por algum número.

Há um ponto curioso aqui porque esta última definição parece incluir a anterior. Afinal, segundo Euclides, um número primo é medido pela unidade. A verdade é que, para os gregos daquela época, a unidade não era considerada um número. No Livro XIV da *Metafísica*, Aristóteles nos diz que 'o um' significa uma unidade de medida, e depois de identificar várias dessas unidades, continua

> 'O um' significa a medida de alguma pluralidade, e 'número' significa uma pluralidade que foi medida e uma pluralidade de medidas. (Assim é natural que um não seja um número; porque a medida não é uma medição, mas ambos a unidade de medida e o um são pontos de partida).

Como veremos na seção 1 do capítulo 1, continuamos excetuando o 1 tanto do domínio dos primos, quanto dos compostos, embora por razões muito diferentes dos gregos de 2300 anos atrás.

Os gregos também chamavam os números compostos de secundários, e a explicação para isso pode ser encontrada nos *Elementos*. Na proposição 31 do mesmo Livro VII, Euclides mostra que todo número pode ser dividido por um primo. Portanto, os números compostos são secundários no sentido de que podem ser construídos a partir dos números primos (ou primários). Hoje em dia expressamos isto de uma maneira muito mais precisa usando o Teorema da Fatoração Única. Contudo, este teorema só foi enunciado e provado por C. F. Gauss, no século XIX. Voltaremos a isto um pouco mais adiante.

Euclides viveu em Alexandria, e trabalhou na grande Biblioteca, que além de repositório de livros, era um importante centro de pesquisas da época; e é

a um outro matemático de Alexandria que caberia o mérito de descobrir o primeiro procedimento prático para achar primos, o *Crivo de Eratóstenes*. Além de matemático, Eratóstenes foi um geógrafo importante que ficou famoso por medir a circunferência da Terra. Por sua proficiência em muitos dos ramos do conhecimento foi apelidado de 'Pentatlos' por seus contemporâneos, em referência ao esporte olímpico em que o atleta participa de cinco provas diferentes. Nicômaco, um matemático que atuou por volta de 100 d. C., descreveu o método de Eratóstenes para achar primos da seguinte maneira:

> O método para obtê-los [os números primos] é chamado por Eratóstenes uma peneira [crivo], porque tomamos os números ímpares misturados de maneira indiscriminada e, por este método, como se fosse pelo uso de um instrumento ou peneira, separamos os primos ou indecomponíveis dos secundários ou compostos.

Portanto, o *crivo* atua como uma espécie de peneira que retém os números primos, mas deixa passar os compostos. Estudaremos o crivo em detalhes na seção 3 do capítulo 1.

Da Grécia Antiga passamos para a Europa do século XIX. Isto não significa que a teoria de números não tenha feito muitos avanços neste ínterim; basta lembrar Diofanto, Fermat e Euler, que serão mencionados mais adiante. Contudo, muito do que foi feito não diz respeito diretamente ao problema que mais nos interessa: como determinar que um número dado seja primo.

Nossa primeira parada é em 1801 quando o jovem C. F. Gauss, então com apenas 24 anos, publicava uma das mais importantes obras matemáticas do século, a *Disquisitiones Arithmeticae*. Foi neste livro que Gauss deu, como já dissemos anteriormente, a primeira demonstração satisfatória do Teorema da Fatoração Única para os inteiros. Do nosso ponto de vista, o outro ponto alto desta obra foi a definição da congruência módulo um inteiro positivo, que desempenha um papel central em quase tudo o que faremos neste livro. A notação para a congruência que usamos hoje em dia é a mesma introduzida por Gauss nas *Disquisitiones*.

O interesse de Gauss pela teoria de números começou muito cedo. Dez anos antes da publicação das *Disquisitones*, e com apenas 14 anos, Gauss escrevera no verso da folha de uma tábua de logaritmos

$$\text{Primzahlen unter } a\,(=\infty)\frac{a}{\text{l}a}.$$

Que uma vez decifrado (e traduzido), significa

> A quantidade de primos (positivos) menores que um inteiro $a > 0$ se aproxima de $a/\ln a$ quando a fica muito grande.

Para entender o significado deste resultado, devemos recuar novamente à Grécia Antiga. Na proposição 20 do Livro IX dos *Elementos*, Euclides prova que existe uma infinidade de primos; veja a seção 5 do capítulo 3 de **[Coutinho 2001]** para mais detalhes. Levando isto em conta, temos que o estoque de

primos menores que um inteiro positivo n aumenta, à medida que n aumenta. Isto nos dá um número progressivamente maior de primos com os quais construir novos números compostos. Portanto, é de esperar que haja intervalos cada vez maiores formados por números compostos entre dois primos consecutivos. Escrevendo $\pi(x)$ para o número de primos positivos menores ou iguais a um número $x > 0$, isto sugere que $\pi(x)$ deve crescer cada vez mais devagar à medida que x aumenta.

Gauss, entretanto, foi ainda mais longe. Ele percebeu que, para valores grandes de x, a função $\pi(x)$ está muito próxima de ser igual a $x/\ln x$; ou, em uma formulação mais precisa

$$\lim_{x\to\infty} \frac{\pi(x)\ln x}{x} = 1.$$

Gauss deve ter chegado a esta conclusão a partir da análise de listas de primos: sabe-se que ele estudou os primos até 3 milhões. Fórmulas semelhantes para o número de primos eram conhecidas de Legendre e Euler.

Como nem Gauss, nem seus antecessores, deram demonstrações para suas fórmulas, o primeiro resultado matematicamente provado sobre a distribuição dos primos só viria com a publicação, em 1848, do artigo *Sur les nombres premiers*, do matemático russo P. L. Chebyshev. Nele, Chebyshev mostrou que

$$0,922 \leq \frac{\pi(x)\ln x}{x} \leq 1,105,$$

para todo $x \geq 3$. Provaremos uma versão muito fraca de uma destas desigualdades na seção 4 do capítulo 1.

É fácil mostrar, a partir do resultado de Chebyshev, que o limite da função $\pi(x)\ln x/x$, quando x tende a infinito, tem que ser 1, *desde que se saiba que o limite existe*. Isto, porém, só seria provado em 1896, independemente por J. Hadamard e C.-J. de La Vallé Poussin. Este resultado é conhecido hoje em dia como o Teorema dos Números Primos. Uma descrição mais detalhada da história deste teorema pode ser encontrada em **[Bateman e Diamond 1996]**.

No artigo 329 das *Disquisitiones* Gauss escreveu que

> O problema de distinguir números primos de compostos, e de resolver números compostos em seus fatores primos, é um dos mais importantes e úteis de toda a aritmética. (...) A dignidade da ciência parece pedir que toda ajuda para a solução de um problema tão elegante e celebrado seja zelosamente cultivada.

Apesar disto, a verdade é que seu livro empurrou a teoria de números em uma direção bem diferente. Nas mãos dos principais sucessores de Gauss, especialmente Kummer e Dedekind, a teoria de números ficou cada vez mais algébrica. Isto se deu, em parte, por influência dos problemas que estes matemáticos abordaram.

O mais famoso destes problemas foi, sem dúvida, o Último Teorema de Fermat. Fermat havia sido o maior expoente da teoria dos números no século

XVII, além de ser um dos precursores do cálculo e da geometria analítica. Um livro que teve grande impacto sobre Fermat foi a (então recém traduzida) *Álgebra* do matemático grego Diofanto, que viveu por volta de 250 d. C.. Fermat escreveu muitos comentários à margem de sua cópia da obra de Diofanto e, após sua morte, seu filho Samuel publicou uma edição da *Álgebra* incluindo as anotações que seu pai havia feito. Na mais famosa destas anotações, Fermat declara ter uma bela demonstração do fato de que se x, y e z são inteiros que satisfazem a equação $x^n + y^n = z^n$, onde $n > 2$, então um destes três números tem que ser zero. "Porém", ele conclui, "esta margem é muito estreita para escrevê-la". Como se sabe o resultado foi finalmente provado por A. Wiles em 1995.

As muitas tentativas frustradas de provar o Último Teorema de Fermat, feitas ao longo dos anos, levaram, em alguns casos, ao desenvolvimento de áreas inteiramente novas da matemática. Foi o que aconteceu com E. Kummer em 1844. A estratégia de Kummer consistia em fatorar o polinômio $x^p + y^p$, quando p fosse primo. Isto só é possível se introduzirmos números complexos. Naturalmente, ao fazer isto, saímos do domínio dos números inteiros e passamos a operar em um subconjunto dos complexos. Este novo conjunto partilha muitas das boas propriedades dos inteiros, entre elas o fato de que cada um de seus elementos pode ser escrito como um produto de elementos irredutíveis, definidos de maneira semelhante aos primos em $\mathbb{Z}$. Infelizmente, a fatoração em irredutíveis nestes conjuntos não é única, e a demonstração inicial de Kummer mostrou-se incorreta. Para restabelecer alguma forma de unicidade para esta fatoração, e salvar sua idéia ao menos em parte, Kummer introduziu o que chamou *ideais*. Generalizado, mais tarde, por Dedekind este se tornou um dos conceitos fundamentais da álgebra moderna, e dedicaremos a ele boa parte do capítulo 3.

Os métodos atualmente utlizados em álgebra surgiram originariamente de duas fontes: a teoria de ideais e a teoria de equações. Esta última estuda os métodos algébricos utilizados para resolver equações polinomiais. Mais precisamente, almeja-se obter uma fórmula explícita para as raízes de uma equação polinomial em função dos coeficientes da equação. Além do mais, esta fórmula deve usar apenas as operações básicas: adição, subtração, multiplicação, divisão e extração de raízes.

Em 1801, ano em que as *Disquisitiones* foram publicadas, o estado desta teoria era um pouco incerto. Desde a Renascença, conheciam-se fórmulas algébricas explícitas se a equação tivesse grau 2, 3 ou 4. Porém, ao analisar e generalizar o método pelo qual estas fórmulas haviam sido obtidas, J. L. Lagrange descobriu, em 1770, que ele não poderia funcionar para equações de grau maior ou igual a 5.

Não tardaria, porém, até que o problema fosse completamente resolvido. O primeiro resultado conclusivo foi a demonstração pelo matemático norueguês N. H. Abel de que uma equação geral de grau maior ou igual a 5 não admite nenhuma fórmula do tipo desejado. Pouco tempo depois, E. Galois daria um critério explícito que pode ser usado para decidir se é ou não possível

expressar as raízes de uma equação em termos dos coeficientes, usando apenas as operações fundamentais.

A idéia central da teoria desenvolvida por Galois é de considerar as várias maneiras pelas quais é possível permutar as raízes de um polinômio sem alterá-lo. Ao conjunto destas permutações Galois deu o nome de *grupo* da equação. Apesar deste nome ter sido introduzido por Galois, a idéia de grupo já existia, em forma latente, tanto no trabalho de Abel quanto no de Lagrange.

A teoria de grupos logo se difundiu para outros ramos da matemática, entre eles a geometria e a teoria de números. A enorme gama de aplicações dos grupos levaria os matemáticos a estudá-los por si mesmos, dando lugar à teoria de grupos, de que trata o capítulo 2. Na verdade, vários teoremas das *Disquisitiones* de Gauss podem ser encarados como uma forma incipiente de resultados mais gerais da teoria de grupos.

O desenvolvimento da álgebra, incluindo a teoria de grupos e o estudo dos ideais, ocupou alguns dos maiores matemáticos do fim do século XIX e início do século XX. Nascidos em grande parte no seio da teoria de números, estes conceitos passaram a ser utilizados para explorar não apenas os inteiros, mas também outros domínios de números com propriedades semelhantes. Enquanto isto acontecia, o estudo da fatoração de inteiros e dos métodos para determinar se um número é primo avançavam, modestamente, nos bastidores.

Um dos maiores expoentes do estudo dos testes de primalidade na segunda metade do século XIX foi o francês F. E. A. Lucas. Ao contrário de outros matemáticos da mesma época, que estudavam ideais e grupos, Lucas nunca obteve um posto universitário. Depois de trabalhar no Observatório de Paris, ele assumiu a posição de professor de Liceu, até a sua morte em 1891.

Por muito tempo a fama de Lucas esteve associada a seu livro *Récréations Mathématiques*, hoje considerado um clássico no assunto. Entretanto, Lucas também descobriu vários métodos para verificar se um número é primo, estabelecendo o princípio básico utilizado por muitos dos testes que usamos atualmente. Seu maior sucesso nesta área foi o teste de primalidade que descobriu para números de Mersenne, e que ele usou para mostrar que $2^{127} - 1$ é primo. Este continua sendo o maior primo descoberto sem o uso de um computador. Refinado por D. H. Lehmer, este teste continua em uso até hoje. Para mais detalhes veja a seção 4 do capítulo 9 de **[Coutinho 2001]**.

Aliás, Lehmer foi um dos pioneiros no uso do computador em teoria de números. Como parte do esforço de guerra entre 1945-46, Lehmer ficou responsável em ajudar na operação do ENIAC (sigla que significa “Electronic Numerical Integrator and Calculator”), utilizado pelas forças armadas americanas para resolver problemas de balística. Nos poucos fins de semana em que o ENIAC estava ocioso, Lehmer e sua esposa, que também era matemática, usavam a máquina noite adentro, para obter informações sobre diversos problemas de teoria de números.

Durante a maior parte do século XX, o estudo dos testes de primalidade continuou sendo apenas um tópico secundário em matemática, e esta situação

só iria mudar a partir da década de 1970. Duas coisas contribuiriam para isso. A primeira, foi o desenvolvimento de novos sistemas de criptografia, como o RSA e o El Gamal, para cuja implementação é necessário ter à disposição um estoque de primos grandes. A segunda, foi a construção de computadores cada vez mais poderosos, que tornaram possível aplicar testes cujo custo computacional excedia, em muito, o que uma pessoa, ou mesmo um grupo de pessoas, podia fazer.

Um dos testes que se tornaram mais populares desde então foi originalmente publicado, em russo, por M. Artjuhov em 1967. Este teste é o aperfeiçoamento de um outro, que já era conhecido desde o século XVII. O ponto de partida de ambos os testes é um resultado de Fermat, segundo o qual,

> se p é um primo, então $a^{p-1} - 1$ é divisível por p, sempre que a for um inteiro que não é divisível por p.

O teste mais antigo consistia em observar que, se descobrirmos um número $0 < a < n$ tal que $a^{n-1} - 1$ não é divisível por n, então n é composto. De fato, se n fosse primo, então não poderia dividir a, já que $a < n$ e, portanto, $a^{n-1} - 1$ teria que ser divisível por n pelo resultado de Fermat.

Alguns matemáticos chegaram a acreditar que a recíproca deste teorema de Fermat seria verdadeira. Se fosse este o caso, bastaria testar se $2^{n-1} - 1$ é divisível por n, para provar que um número $n > 2$ é primo. Entretanto isto é falso. Como veremos no capítulo 4, mesmo que $a^{n-1} - 1$ seja divisível por n para muitos valores diferentes de a não podemos garantir que n seja primo.

O teste de Artjuhov foi redescoberto dez anos depois por J. Selfridge, e popularizado pelos trabalhos de G. Miller e M. Rabin. Hoje em dia este teste é a base da maioria dos métodos "rápidos" pelos quais os sistemas de computação algébrica certificam que um número é primo. Entretanto, como veremos no capítulo 4, o método de Artjuhov não garante que um número é primo. Na versão de Rabin, o teste apenas certifica que há uma probabilidade P de que o número seja primo. Contudo, podemos refinar o teste de modo a fazer P tão pequena quanto desejarmos, ainda que há um custo computacional maior.

Com isso nos deparamos com a questão de quanto custa mostrar que um dado número é primo. Do ponto de vista prático, a situação, ao final do século XX, apresentava dois grupos diferentes de testes. De um lado havia os testes rápidos, mas que só garantem que o número é "provavelmente primo", ao estilo de Artjuhov e Rabin. De outro, havia os testes mais lentos, mais que provavam primalidade, como o que foi proposto por Lucas em 1876. Este teste é baseado no fato de que

> se $a > 0$ e $n > 1$ são inteiros tais que $a^{n-1} \equiv 1 \pmod{n}$, mas $a^{(n-1)/q} \not\equiv 1 \pmod{n}$ para todo primo q que divide $n - 1$, então n é primo.

O problema em aplicar este teste, e outros como ele, é que, além de ter que achar um a adequado, precisamos ser capazes de fatorar $n - 1$ completamente o que, geralmente, só pode ser feito a um custo muito alto.

O estudo da complexidade dos algoritmos, que tomou forma com os trabalhos de J. Hartmanis e R. E. Stearns, a partir dos anos 1960, permitiu categorizar estes dois tipos de testes de uma maneira mais conceitual. Assim, temos de um lado os algoritmos cujo tempo de execução é uma função polinomial do tamanho da entrada. A esta categoria pertence o teste de Rabin. De outro, temos os testes cujo tempo de execução é uma função exponencial da entrada. Um teste de primalidade exponencial bastante elementar é baseado no seguinte resultado, conhecido como teorema de Wilson,

$$n \text{ é primo se, e somente se, } (n-1)! \equiv -1 \pmod{n}.$$

Neste caso, o problema é que não conhecemos algoritmos polinomiais para calcular $(n-1)!$ módulo n.

Estes testes estão longe de esgotar a enorme variedade de métodos utilizados para verificar que um número é primo, e que incluem, por exemplo, o teste da curva elíptica e o teste proposto por Adleman, Pomerance and Rumely em 1983. Este último tem a distinção de ser um algoritmo subexponencial; isto é, seu tempo de execução é uma função que, apesar de não ser polinomial, não cresce tão rapidamente quanto uma exponencial.

Era este, essencialmente, o estado das coisas em agosto de 2002 quando Manindra Agrawal, Neeraj Kayal e Nitin Saxena divulgaram seu artigo na web. Chamado de *Primes is in P*, o artigo detalha um algoritmo de primalidade que não apenas prova que um número dado é primo (ou composto), mas que tem um tempo de execução que é uma função polinomial da entrada. O impacto foi imediato, até porque o teste não é apenas muito fácil de descrever, mas seu funcionamento depende apenas de métodos algébricos elementares (grupos, anéis e corpos) bem conhecidos de qualquer aluno de graduação em matemática. O único resultado realmente sofisticado utilizado no artigo de Agrawal, Kayal e Saxena dizia respeito à distribuição de primos em um intervalo, e era necessário, apenas, para mostrar que o algoritmo operava em tempo polinomial.

Não tardou para que o algoritmo fosse modificado, melhorado e generalizado. Passados dois anos da divulgação do artigo na web, já há versões do algoritmo, agora conhecido como AKS, que utilizam apenas resultados elementares, mesmo para a análise do custo do algoritmo. Além disso, há o entendimento de que este é o primeiro de uma nova classe de algoritmos, conhecidos como algoritmos ciclotômicos ou do tipo AKS.

O objetivo deste livro é dar uma descrição completa e detalhada de uma versão do algoritmo AKS devida a H. W. Lenstra Jr. Assumimos como pré-requisito apenas um conhecimento básico de teoria de números, ao nível de **[Coutinho 2001]**. Todo o resto será exposto aqui.

No capítulo 1 tratamos de uma miscelânea de resultados básicos sobre primos que vão, desde a definição de primo e o Teorema da Fatoração Única na seção 1, até uma versão muito simplificada do resultado de Chebyshev sobre a distribuição de primos, na seção 4. A seção 2 é dedicada ao estudo da complexidade dos algoritmos utilizados em teoria de números, e os conceitos que lá

introduzimos são usados na seção 3 para fazer uma análise do custo do Crivo de Eratóstenes.

Os dois capítulos seguintes são dedicados ao estudo das estruturas algébricas mais elementares: grupos abelianos no capítulo 2, e anéis e corpos no capítulo 3. Isto corresponde, como vimos acima, ao início do desenvolvimento da álgebra moderna no século XIX. A maior parte do capítulo 2 é mera revisão do capítulo 7 de **[Coutinho 2001]**. A principal diferença está na seção 4, em que introduzimos os grupos quocientes, que desempenham um papel importante na demonstração do algorimo AKS. Já o capítulo 3 é uma introdução aos elementos da teoria de anéis, com ênfase nos anéis de polinômios. Neste capítulo estabelecemos, também, os conceitos de ideal e de anel quociente, e provamos o Teorema da Fatoração Única para polinômios em uma variável sobre um corpo. Estes dois capítulos podem ser omitidos, caso você já tenha conhecimentos de álgebra equivalentes ao do final de um curso de graduação em matemática.

O capítulo 4 prepara o terreno para o AKS, descrevendo não apenas alguns dos resultados que usaremos na demonstração daquele algoritmo, como também o teste de primalidade probabilístico de Rabin, mencionado acima. Finalmente, o capítulo 5 é completamente dedicado ao algoritmo AKS. Além de descrevê-lo em detalhes, provamos que funciona corretamente e que seu tempo de execução é uma função polinomial da entrada.

Curiosamente, uma das etapas do AKS foge completamente à linha de resultados de teoria de números e álgebra que formam a espinha dorsal deste livro. Isto ocorre porque o resultado fundamental do qual depende o funcionamento do AKS garante apenas que o número que está sendo testado é potência de um primo. Contornamos este problema verificando se o número dado é uma potência de inteiro, antes mesmo de aplicar o algoritmo propriamente dito. Para isto usamos uma variação (módulo 2) do método inventado por Newton para achar raízes de uma equação. Este procedimento é discutido em detalhes no apêndice.

Finalmente, o epílogo contém algumas considerações finais sobre primalidade e o algoritmo AKS, incluindo referências sobre questões relativas à sua implementação.

Capítulo 1

Primos

Neste capítulo relembramos o que significa um número ser primo e discutimos o mais antigo dos métodos utilizados para encontrar primos, o *crivo de Eratóstenes*. Apesar de ser muito fácil de programar, o crivo não é adequado para encontrar primos grandes. Para poder explicar o porquê desta última afirmação, formalizamos na seção 2 a noção de custo de execução de um algoritmo. A última seção é dedicada a alguns resultados elementares sobre a distribuição dos primos que serão utilizados para estudar o custo do algoritmo AKS, no capítulo 5.

1. Primos

Dizemos que um número inteiro não nulo $p \neq \pm 1$ é *primo*, se os únicos fatores de p são ± 1 e $\pm p$. Um número inteiro não nulo $n \neq \pm 1$, que não é primo, é chamado de *composto*.

Observe que 0 e ± 1 não são compostos, nem primos. Ao passo que a exclusão do 0 parece natural, por causa das propriedades peculiares do 0 com respeito à divisão, o mesmo não se pode dizer de ± 1. Afinal, não podemos fatorar 1, nem -1, como produto de dois inteiros menores (em valor absoluto!), portanto eles deveriam figurar como primos. A razão pela qual isto não acontece está ligada ao seguinte teorema.

TEOREMA DA FATORAÇÃO ÚNICA. *Dado um inteiro positivo $n \geq 2$ podemos sempre escrevê-lo, de modo único, na forma*

$$n = p_1^{e_1} \dots p_k^{e_k},$$

onde $1 < p_1 < \cdots < p_k$ são números primos e $e_1, \dots, e_k$ são inteiros positivos.

Observe que o teorema nos diz duas coisas. Em primeiro lugar, uma fatoração na forma do teorema *existe*; em segundo, ela é *única*. Para entender o significado da unicidade da fatoração, imagine a seguinte situação. Distribuímos um certo inteiro $n \geq 2$ entre várias pessoas, e pedimos que decomponham n na forma dada pelo teorema. Mesmo deixando estas pessoas livres para escolherem o método pelo qual chegarão à decomposição desejada, a unicidade da fatoração garante que *todos encontrarão a mesma decomposição*.

Podemos, agora, explicar porque 1 e -1 não podem ser contados entre os números primos. O problema é que, se fizermos isto, perderemos a unicidade da fatoração no teorema acima. Por exemplo, a única maneira de fatorar 12 na forma do teorema é escrever $12 = 2^2 \cdot 3$. Mas se 1 e -1 fossem primos, teríamos muitas outras fatorações distintas possíveis, como

$$12 = 1 \cdot 2^2 \cdot 3 = 1^2 \cdot 2^2 \cdot 3 = 1^5 \cdot 2^2 \cdot 3,$$

e assim por diante. Na verdade, cada número composto admitiria uma infinidade de fatorações distintas. Tendo em vista a enorme utilidade da unicidade da fatoração, excluir ± 1 dentre os primos parece um pequeno preço a pagar em vista de um bem maior. Voltaremos a discutir o papel dos números ± 1 quando estudarmos a fatoração de polinômios no capítulo 3.

Como vimos na introdução, este teorema também oferece uma explicação da razão pela qual os números primos têm este nome. Afinal, estes números podem ser usados como tijolos na construção de todos os números inteiros. Portanto, os números primos são aqueles que "vêm primeiro", no sentido de serem mais fundamentais.

Dada a importância do Teorema da Fatoração Única, não podemos passar adiante sem comentar sobre a sua demonstração. Na verdade, há duas coisas distintas a provar: a existência e a unicidade da fatoração. Ao passo que um argumento bastante elementar é suficiente para provar a existência da fatoração, a unicidade é muito mais sutil. É por isto que, embora os matemáticos venham assumindo a unicidade da fatoração desde a Grécia Antiga, uma demonstração satisfatória deste fato só foi dada por C. F. Gauss no século XIX. Relembramos, a seguir, o algoritmo de tentativa e erro geralmente utilizado para determinar a fatoração de um número. A demonstração da unicidade da fatoração aparece como o exercício 13 do capítulo 3. Veja também **[Coutinho 2001, capítulo 2, seção 8]**.

Começaremos descrevendo um algoritmo bastante ingênuo, que encontra apenas um fator de um inteiro positivo n dado. O algoritmo simplesmente tenta dividir n por cada um dos inteiros de 2 a $n-1$. Se algum destes inteiros dividir n, então achamos um fator de n. Se nenhum fator menor que n for encontrado, então n é primo.

Uma característica notável deste algoritmo, é que sempre encontra um fator *primo* de n. Para entender o porquê desta afirmação, suponhamos que n seja composto e que $f \geq 2$ seja o fator de n encontrado pelo algoritmo. Se $f' > 1$ for um fator primo de f, então pela definição de divisibilidade, existem inteiros a e b tais que

$$n = fa \quad \text{e} \quad f = f'b.$$

Logo $n = f' \cdot ab$. Contudo, o algoritmo varre os inteiros, do menor ao maior, à procura de fatores, de modo que f será sempre o *menor* fator positivo de n. Como já vimos que f' também é fator de n, concluímos que $f' \geq f$. Entretanto, como f' divide f, temos também que $1 < f' \leq f$. Mas isto só é possível se $f' = f$; provando assim que f é primo.

Há ainda uma outra observação que precisamos fazer, e que leva a uma melhora da performance do algoritmo. Conforme descrevemos acima, o algoritmo apenas executa uma busca de fatores, começando de 2. Se nenhum fator menor que n for encontrado, então o número é primo. Contudo, como o fator de um número composto encontrado pelo algoritmo sempre é o menor possível, podemos parar de executá-lo muito antes de chegar a $n-1$. De fato se n for composto e seu menor fator primo for f, então existe um inteiro positivo a tal que $n = fa$. Como a também é fator de n, temos ainda que $f \leq a$. Substituindo esta desigualdade em $n = fa$, obtemos

$$n = fa \geq f^2, \text{ donde } \; f \leq \sqrt{n}.$$

Isto significa que, se n for composto, então seu menor fator primo será menor ou igual a $\sqrt{n}$. Portanto, se o algoritmo não encontrar nenhum fator primo até a raiz quadrada do número ao qual está sendo aplicado, então este número é primo.

Resumindo o que vimos até agora: o algoritmo deve buscar um número que divida n, começando de 2 e avançando, de um em um, até $\sqrt{n}$. Se n for composto, encontraremos o menor fator de n por este método, e este fator é *necessariamente primo*. Se nenhum dos números pesquisados é fator de n, isto significa que n é primo. Reunindo tudo o que já fizemos, podemos descrever as etapas do algoritmo como segue.

Algoritmo de fatoração

Entrada: inteiro positivo $n \geq 2$.
Saída: inteiro positivo f que é o menor fator primo de n ou uma mensagem indicando que n é primo.

Etapa 1: Comece fazendo $F = 2$.
Etapa 2: Se $F \geq \sqrt{n}$ escreva n *é primo* e páre.
Etapa 3: Se n/F é inteiro escreva 'F *é fator de* n' e páre; senão, incremente F de uma unidade e volte à Etapa 2.

De posse deste algoritmo, podemos encontrar a fatoração completa de qualquer inteiro positivo na forma dada pelo teorema. Suponhamos que $n > 2$ é composto. Aplicando o algoritmo descrito acima a n, encontramos seu menor fator, que chamaremos de q_1. A seguir aplicamos o algoritmo novamente, só que desta vez ao co-fator n/q_1. Se n/q_1 for primo já achamos a fatoração completa de n. Se não for, teremos determinado um segundo fator primo de n, que vamos chamar de q_2. É claro que $q_1 \leq q_2$, mas pode acontecer que sejam iguais, basta que q_1^2 divida n. Continuando, aplicamos o algoritmo ao co-fator $n/(q_1 q_2)$, e assim por diante. Como a seqüência dos co-fatores

$$n > \frac{n}{q_1} > \frac{n}{q_1 q_2} > \frac{n}{q_1 q_2 q_3} > \cdots > 0$$

é decrescente, este procedimento vai ter que parar em algum momento. Quando isto acontecer teremos determinado a fatoração completa de n em primos.

O problema deste algoritmo é que ele é extremamente ineficiente; isto é, ele leva muito tempo para achar um fator se aplicado a um número relativamente grande. Mas como se mede o tempo de execução de um algoritmo de maneira precisa? Este é o tema da próxima seção, ao final da qual analisamos, em detalhe, como se comporta o tempo de execução do algoritmo de fatoração descrito acima.

2. Custo de um algoritmo

Nesta seção tratamos de como medir o custo, ou tempo de execução, de um algoritmo. Esta é uma área enorme da teoria de algoritmos, e não pretendemos discutir mais do que o mínimo necessário para nossas necessidades neste livro.

Uma boa medida do custo de um algoritmo do tipo que estamos considerando é a quantidade de operações aritméticas elementares que são realizadas ao executá-lo. Por *operação aritmética elementar* entendemos uma operação efetuada sobre algarismos, seja ela adição, subtração, multiplicação, divisão ou cálculo do resto. Além disso, como desejamos implementar os algoritmos em um computador, estaremos pensando em termos de algarismos binários, ou *bits*.

Naturalmente, a quantidade de operações aritméticas elementares que são executadas depende do tamanho da entrada que é fornecida ao algoritmo. Por exemplo, para calcular o dobro de 1000 precisamos realizar mais operações do que para calcular o dobro de 100.

Começaremos por analisar quantas operações são necessárias quando somamos dois inteiros. Sejam

$$a = (a_{k-1}, \ldots, a_1, a_0)_2 \text{ e } b = (b_{k-1}, \ldots, b_1, b_0)_2,$$

as representações binárias dos inteiros a e b. Isto significa que cada um dos números $a_0, \ldots, a_{k-1}, b_0, \ldots, b_{k-1}$ é 0 ou 1 e que

$$a = a_{k-1}2^{k-1} + \cdots + a_1 2 + a_0 \text{ e } b = b_{k-1}2^{k-1} + \cdots + b_1 2 + b_0.$$

Para efetuar a soma da maneira usual, começamos calculando $a_0 + b_0$. Se este número for menor que 2, então $s_0 = a_0 + b_0$ é o algarismo das unidades de $a + b$. Por outro lado, se $a_0 + b_0 = 2$, então $s_0 = 0$ e vai 1. O tanto que 'vai' para a próxima etapa é conhecido como *reserva*. Seja como for, efetuamos até aqui apenas uma operação aritmética elementar.

Na próxima etapa cacularemos $a_1 + b_1$ se a reserva é 0, ou $a_1 + b_1 + 1$, se houve reserva não nula. Já aqui a coisa se complica, porque teremos que realizar uma ou duas operações aritméticas elementares, dependendo se a reserva é ou não nula. Daí em diante o processo segue o mesmo padrão.

Quantas operações aritméticas elementares realizamos, afinal de contas? É difícil dizer exatamente, porque o resultado depende de quantas reservas não nulas apareceram ao longo da conta, e isto varia de número para número. Entretanto, é claro que não precisamos efetuar mais do que $2k$ operações aritméticas elementares, neste caso.

Antes de continuar precisamos chamar a atenção para o fato de que há muitas outras operações que um computador tem que realizar para efetuar uma soma, e que não foram incluídas entre o que chamamos de operações aritméticas elementares. Uma destas é a comparação de bits; usada, por exemplo, para ver se a reserva é zero. Mas há outras, algumas que sequer dizem respeito a números inteiros, como é o caso do acesso à memória. Apesar destas operações afetarem o tempo que o computador leva para executar um dado algoritmo, vamos ignorá-las e nos concentrar apenas no número de operações aritméticas elementares como as definimos. Uma razão para isto é que este número é menos dependente da implementação que foi feita para o algoritmo, ou do computador onde o programa está rodando.

Passemos à multiplicação. Quantas operações aritméticas elementares serão necessárias se quisermos multiplicar a por b? Utilizando o algoritmo usual, precisamos multiplicar primeiro b por cada bit de a e depois somar os resultados. Entretanto, ao multiplicar b por um bit a_i de a efetuamos não apenas produtos de bits, mas também somas, porque pode haver reserva nos produtos! É claro que teremos exatamente k produtos elementares. Por outro lado, como não pode haver mais do que $k-1$ reservas, teremos no máximo $k-1$ somas. Assim, ao multiplicar b por a_i não precisamos de mais do que $2k-1$ operações aritméticas elementares. Logo, para calcular o produto de b por cada um dos bits de a, teremos que realizar, no máximo, $k(2k-1)$ operações aritméticas elementares.

Só que, para calcular ab precisamos somar estes produtos de b por bits de a, sem esquecer a ordem de grandeza de cada bit! Em outras palavras, se

$$ba_i = (c_{i,k+1}, \ldots, c_{i,1}, c_{i,0})_2$$

então a soma que vamos calcular pode ser escrita na forma

$$\begin{array}{cccccc}
 & & & c_{0,k+1} & \cdots & c_{0,1} & c_{0,0} \\
 & & c_{1,k+1} & \cdots & c_{1,1} & c_{1,0} & \\
 & c_{2,k+1} & \cdots & c_{2,1} & c_{2,0} & & \\
 & \vdots & \vdots & \vdots & & & \\
c_{k,k+1} & \cdots & c_{k,0} & & & &
\end{array}$$

O número de somas realizadas em cada coluna não pode exceder $2k$, mesmo levando em conta as reservas. Como temos k colunas, podemos somar todos estes números a um custo de $2k^2$ somas elementares. Juntando tudo verificamos que, ao calcular ab, não precisamos realizar mais do que $4k^2 - k$ operações aritméticas elementares.

Você já deve ter notado que, para determinar o número aproximado de operações aritméticas elementares realizadas para calcular ab fizemos muito mais concessões do que quando tratamos da soma. O fato é que teremos que fazer mais e mais concessões do tipo “não pode haver mais do que” à medida que as operações que estamos analisando forem se complicando. Por isso, a

prática corrente é de ignorar os detalhes da expressão obtida para a quantidade de operações aritméticas elementares e se concentrar apenas na maior potência de k que aparece na expressão desta quantidade.

Formalizamos isto usando a notação O. Sejam f e g duas funções cujo contradomínio é $\mathbb{R}$, e cujo domínio é $\mathbb{N}$ ou $\mathbb{R}$. Dizemos que $f(x)$ *é* $O(g(x))$ se existem constantes C_1 e C_2 tais que $f(x) \leq C_1 g(x)$ sempre que $x > C_2$. Um exemplo simples, mas que usaremos com freqüência, é o seguinte. Seja $f(x)$ um polinômio de grau n com coeficients reais. Então, $f(x)$ é $O(x^n)$. Para ver que isto é verdade, digamos que $f(x) = a_n x^n + \cdots + a_0$, onde $a_n, \ldots, a_0$ são números reais e $a_n \neq 0$. Escolha um número real $b > a_n$. Como

$$\lim_{x\to\infty} \frac{f(x)}{bx^n} = \frac{a_n}{b} < 1,$$

concluímos que, sempre que x for grande o suficiente, teremos $f(x) \leq bx^n$. Mas isto significa que $f(x)$ é $O(x^n)$, como havíamos dito.

As seguintes propriedades da notação O são facilmente verificadas, e serão usadas a seguir sem maiores comentários; veja exercício 3. Suponha que f_1, f_2, g_1 e g_2 são funções cujo contradomínio é $\mathbb{R}$, e cujo domínio é $\mathbb{N}$ (ou $\mathbb{R}$). Se f_1 é $O(g_1)$ e f_2 é $O(g_2)$, então

(1) $f_1 + f_2$ é $O(\max\{g_1, g_2\})$, e
(2) $f_1 f_2$ é $O(g_1 g_2)$.

Em particular, se tanto f_1, quanto f_2, são $O(g_1)$, então $f_1 + f_2$ é $O(g_1)$ e $f_1 f_2$ é $O(g_1^2)$.

Usando esta notação, a soma de dois números de k bits tem custo $O(k)$ e a multiplicação tem custo $O(k^2)$. É igualmente fácil ver que a subtração tem custo $O(k)$, mas é muito mais difícil mostrar que a divisão pode ser realizada ao custo $O(k^2)$. Para mais detalhes veja **[von zur Gathen e Gerhard 1999, p. 38]**. Como um inteiro positivo n tem, aproximadamente, $\log_2 n$ bits, podemos dizer que a soma de dois inteiros menores ou iguais a n pode ser calculada ao custo $O(\log_2 n)$. Já a multiplicação e a divisão têm custo $O((\log_2 n)^2)$.

Muitos dos cálculos que faremos ao longo deste livro terão lugar em $\mathbb{Z}_n$, e não em $\mathbb{Z}$. Por isso precisamos determinar o custo das operações com classes módulo n. Sejam $\overline{a}, \overline{b} \in \mathbb{Z}_n$. Definimos a adição e a multiplicação de classes por

$$\overline{a} + \overline{b} = \overline{a+b} \text{ e } \overline{a} \cdot \overline{b} = \overline{ab}.$$

Ou seja, primeiro efetuamos a operação correspondente com os representantes das classes em $\mathbb{Z}$, depois reduzimos o resultado módulo n. Supondo que $0 \leq a, b \leq n-1$, podemos calcular $a+b$ ao custo $O(\log_2 n)$ e ab ao custo $O((\log_2 n)^2)$. Para determinar o resíduo módulo n de $a+b$ e de ab, devemos dividi-los por n e tomar o resto. Como $a+b \leq 2n$ e $ab \leq n^2$, podemos dividi-los por n a um custo de $O((\log_2 n)^2)$. Portanto, o custo total tanto da adição, quanto da multiplicação, em $\mathbb{Z}_n$ será $O((\log_2 n)^2)$.

Encerramos a seção aplicando o que aprendemos para determinar o custo do algoritmo da seção 1 que acha um fator primo de um número $n > 1$. Em primeiro lugar precisamos determinar as operações aritméticas que o algoritmo

efetua. Uma olhada, mesmo que superficial, na descrição do algoritmo mostra que há duas operações: uma divisão na etapa 2 e uma soma na etapa 3. Além disso, cada uma destas operações é executada apenas uma vez, a cada laço que o algoritmo completa. Como os números que estão sendo somados são menores que $\sqrt{n}$, a soma executada em cada laço terá custo $O(\log_2 n)$. Já a divisão tem custo $O((\log_2 n)^2)$ porque o dividendo é sempre n. Portanto, cada laço tem um custo da ordem de $O((\log_2 n)^2)$.

Falta ainda estimar o número de laços a serem executados. O pior caso possível ocorre quando n é primo. Neste caso, o algoritmo executa $\lfloor\sqrt{n}\rfloor$ laços a um custo de $O((\log_2 n)^2)$ cada um, o que dá um custo total da ordem de $O(n^{1/2}(\log_2 n)^2)$. Lembre-se que, se a for um número real, então $\lfloor a \rfloor$ denota a parte inteira de a.

Comparado aos algoritmos analisados anteriormente isto é muito ruim. De fato, temos neste caso uma função exponencial do número de bits, já que $n^{1/2} = 2^{(\log_2 n)/2}$. Por causa disso, o número de operações aumenta muito rapidamente quando o número de bits da entrada cresce, o que torna o algoritmo muito lento. Por isso dizemos que o algoritmo de fatoração da seção anterior é ineficiente: o custo em executá-lo é muito alto.

Podemos formalizar a diferença entre um algoritmo eficiente e um que não é eficiente, da seguinte maneira. Em primeiro lugar, vamos continuar medindo o custo de um algoritmo em função do número de bits da entrada n que, como já vimos, é aproximadamente igual a $\log_2 n$. Seja, então, $\mathcal{A}$ um algoritmo e n um inteiro positivo que é entrada de $\mathcal{A}$. Dizemos que $\mathcal{A}$ é um algoritmo *polinomial* se o número de operações aritméticas elementares que ele executa é $O((\log_2 n)^r)$, para algum inteiro $r > 0$. Por outro lado, $\mathcal{A}$ é um algoritmo *exponencial* se executa um número de operações aritméticas elementares que é de ordem $O(2^{\log_2(n)}) = O(n)$, ou pior. Usando esta terminologia, um algoritmo é eficiente se é polinomial, porque um algoritmo exponencial vai ficando muito lento à medida que n cresce. Portanto, algoritmos polinomiais são rápidos, e algoritmos exponenciais são lentos.

Esta seção apenas toca de raspão em uma área enorme, que estuda a complexidade dos algoritmos. Por exemplo, discutimos a eficiência dos algoritmos apenas em função do pior caso possível. Só que, para alguns algoritmos, o pior caso possível raramente ocorre, e uma medida mais realística é dada pela análise de como o algoritmo se comporta em média. Também ignoramos o uso de memória pelos algoritmos. Mas este pode ser um dado importante, como acontece, por exemplo, com o crivo de Eratóstenes que será o tema de nossa próxima seção. Para mais detalhes sobre a análise da complexidade de algoritmos aplicada à teoria de números veja **[Szwacfiter 1986]**, **[Cormen, Leiserson e Rivest 1999, capítulo 33]**, **[von zur Gathen and Gerhard 1999]** ou **[Knuth 1999]**.

3. Crivo de Eratóstenes

O *crivo de Eratóstenes* é o mais antigo dos métodos para achar primos, e não envolve nenhuma fórmula explícita. Como vimos na introdução, ele funciona como uma espécie de peneira que, se aplicada aos inteiros positivos menores que $n > 2$, deixa passar os números compostos, mas retém os primos. Com isso, obtemos a lista completa dos primos positivos menores que n. Vejamos, em mais detalhes, como se faz isto.

Para executar o crivo com lápis e papel podemos proceder da seguinte maneira. Listamos os ímpares de 3 a n. É claro que só listamos os ímpares porque 2 é o único primo par. Começamos então a operar com o crivo propriamente dito. O primeiro número da nossa lista é 3: riscamos os demais números da lista, de 3 em 3. Assim serão riscados todos os múltiplos de 3, maiores que ele próprio. Em seguida procuramos o menor elemento da lista, maior que 3, que não tenha sido riscado, que é 5. Riscamos os demais números da lista, de 5 em 5. Com isso serão riscados todos os múltiplos de 5 maiores que ele próprio. E assim por diante, até chegar a n.

Por exemplo, se $n = 80$, a lista de números é

	3	5	7	9	11	13	15	17	19
21	23	25	27	29	31	33	35	37	39
41	43	45	47	49	51	53	55	57	59
61	63	65	67	69	71	73	75	77	79

Ao final da primeira passagem do crivo (de 3 em 3), ficamos com

	3	5	7	$\cancel{9}$	11	13	$\cancel{15}$	17	19
$\cancel{21}$	23	25	$\cancel{27}$	29	31	$\cancel{33}$	35	37	$\cancel{39}$
41	43	$\cancel{45}$	47	49	$\cancel{51}$	53	55	$\cancel{57}$	59
61	$\cancel{63}$	65	67	$\cancel{69}$	71	73	$\cancel{75}$	77	79

Ao final da segunda passagem do crivo (de 5 em 5) a lista é

	3	5	7	$\cancel{9}$	11	13	$\cancel{15}$	17	19
$\cancel{21}$	23	$\cancel{25}$	$\cancel{27}$	29	31	$\cancel{33}$	$\cancel{35}$	37	$\cancel{39}$
41	43	$\cancel{45}$	47	49	$\cancel{51}$	53	$\cancel{55}$	$\cancel{57}$	59
61	$\cancel{63}$	$\cancel{65}$	67	$\cancel{69}$	71	73	$\cancel{75}$	77	79

Ao final da terceira passagem do crivo (de 7 em 7), a lista se torna

	3	5	7	$\cancel{9}$	11	13	$\cancel{15}$	17	19
$\cancel{21}$	23	$\cancel{25}$	$\cancel{27}$	29	31	$\cancel{33}$	$\cancel{35}$	37	$\cancel{39}$
41	43	$\cancel{45}$	47	$\cancel{49}$	$\cancel{51}$	53	$\cancel{55}$	$\cancel{57}$	59
61	$\cancel{63}$	$\cancel{65}$	67	$\cancel{69}$	71	73	$\cancel{75}$	$\cancel{77}$	79

Já ao final da quarta passagem do crivo (de 11 em 11), a lista continua a mesma acima. A quinta passagem seria de 13 em 13, mas novamente nada vai mudar na lista. Na verdade nenhuma passagem posterior do crivo vai eliminar nenhum número adicional desta lista. Logo os primos ímpares positivos menores que

80 são

$$\begin{array}{cccccccccc} 3 & 5 & 7 & 11 & 13 & 17 & 19 & 23 & 29 & 31 \\ 37 & 41 & 47 & 53 & 59 & 61 & 67 & 71 & 73 & 79. \end{array}$$

Este exemplo nos leva a observar algumas características do crivo. Em primeiro lugar, alguns números são riscados da lista mais de uma vez. É o caso de 15 que já havia sido riscado na primeira passagem (de 3 em 3), e foi riscado também na segunda (de 5 em 5). Em segundo lugar, já havíamos riscado da lista todos os números compostos na terceira passagem do crivo. Todas as passagens posteriores foram redundantes.

Infelizmente não é possível evitar completamente o fato de que alguns números serão riscados várias vezes. Apesar disto, podemos melhorar um pouco o procedimento acima. Digamos que queremos achar os primos até n, e que estamos prestes a riscar os números de p em p, para algum primo p. É claro que os múltiplos de p, que também são múltiplos de primos menores que p, já foram riscados da lista. Portanto, nesta etapa, podemos começar a riscar de p em p a partir do menor múltiplo de p, *que não é múltiplo de um primo menor que* p; isto é, a partir de p^2. Em particular, se estamos para riscar de p em p e se $p > \sqrt{n}$, então o menor composto que ainda não foi riscado em uma etapa anterior seria $p^2 > n$, que está fora da tabela. Portanto, não há necessidade de riscar de p em p para $p > \sqrt{n}$. No exemplo acima $\lfloor \sqrt{80} \rfloor = 8$; por isso bastou riscar de 3 em 3, de 5 em 5 e de 7 em 7 para eliminar todos os compostos menores que 80.

Resumindo: podemos riscar de p em p a começar de p^2, e podemos parar de riscar quando $p^2 > n$. Isto evita algumas duplicações e torna o crivo um pouco mais econômico.

Encerramos esta seção determinando o custo do crivo de Eratóstenes. Digamos que o crivo está sendo usado para achar os primos menores que um inteiro positivo n. Precisamos identificar quais as operações aritméticas que estamos efetuando ao executar o crivo, para podermos contá-las. Analisando a descrição do crivo com cuidado, verificamos que a única operação aritmética executada é a soma. Por exemplo, se vamos riscar os primos de p em p, precisamos calcular $3p, 5p, 7p, \ldots$. Fazemos isto somando $2p$ sucessivamente a p, a $3p$, a $5p$, e assim por diante. Como os números que estão sendo somados são menores que n, cada soma tem custo inferior a $O(n)$.

Para completar a análise precisamos determinar quantas somas estão sendo efetuadas. Mas fazemos uma soma para cada múltiplo ímpar de p menor que n. Logo basta saber quantos destes múltiplos existem entre 1 e n. Entretanto, para facilitar as contas mais adiante, contaremos apenas o número total de múltiplos de p entre 1 e n. Este número é aproximadamente o dobro do número de múltiplos ímpares, mas a diferença desaparece sob a notação O.

Trataremos um caso mais geral, porque o mesmo problema vai sugir na próxima seção. Seja $k > 0$ um número inteiro. Se $kt \leq n$ para algum inteiro positivo t, então $t \leq \lfloor n/k \rfloor$. Portanto, os múltiplos de k menores ou iguais a n são da forma

$$kt \text{ onde } \; k = 1, 2, \ldots, \lfloor n/k \rfloor.$$

Em outras palavras, existem exatamente $\lfloor n/k \rfloor$ múltiplos positivos de k menores que n.

Suponhamos, mais uma vez, que estamos riscando de p em p. Já sabemos que existem $\lfloor n/p \rfloor$ múltiplos de p menores que n. Somando um p de cada vez, começando em p até chegar a n, precisaremos efetuar, no máximo, $\lfloor n/p \rfloor$ somas para determiná-los todos. Como isto vai ser feito para cada um dos primos menores ou iguais a $\sqrt{n}$, concluímos que teremos de executar, no máximo,

$$\sum_{p \in \mathcal{P}(\sqrt{n})} \lfloor n/p \rfloor \leq \sum_{p \in \mathcal{P}(\sqrt{n})} n/p$$

somas, onde $\mathcal{P}(k)$ denota o conjunto dos primos menores ou iguais a k. Entretanto,

$$\sum_{p \in \mathcal{P}(\sqrt{n})} n/p \leq \sum_{p \in \mathcal{P}(\sqrt{n})} n \leq n\sqrt{n} = n^{3/2}.$$

Logo, não precisamos efetuar mais do que $n^{3/2}$ somas, a um custo máximo de $O(\log_2 n)$ cada uma, o que dá um custo total de $O(n^{3/2} \log_2 n)$. Como $\log_2 n \leq \sqrt{n}$ se $n \geq 4$, então o custo do crivo de Eratóstenes é inferior a $O(n^2)$.

Fiéis a nosso lema de manter o livro elementar, fizemos a análise mais simples possível do custo do crivo que fosse compatível às necessidades do algoritmo AKS. Entretanto, isto produz um resultado bem pior do que o obtido por uma estimativa mais cuidadosa. De fato, é possível mostrar **[Hardy e Wright 1994, Theorem 427, p. 351]** que

$$\sum_{p \in \mathcal{P}(\sqrt{n})} n/p = n \ln \ln n + O(n).$$

Usando esta estimativa, verificamos que o custo do crivo é $O(n \ln \ln n \log_2 n)$. Apesar deste resultado ser melhor do que o $O(n^{3/2} \log_2 n)$ obtido anteriormente, vemos que se trata de um algoritmo exponencial. Contudo, o maior problema que se enfrenta ao executar o crivo em um computador não é tanto o tempo de execução, mas a enorme quantidade de memória que ele consome. Para mais detalhes veja **[Crandall e Pomerance, seção 3.2.1]**.

4. Distribuição dos primos

A análise do custo de uma das etapas do algoritmo AKS depende, de maneira crucial, de como os primos se distribuem. Como vimos na introdução, esta é uma área importante, e nada fácil, da teoria de números. Nesta seção provamos um resultado totalmente elementar sobre a distribuição de primos que é suficiente para mostrar que o algoritmo AKS é polinomial.

Para simplificar a demonstração do teorema, isolamos do argumento um lema referente aos números binomiais.

LEMA 1.1. *Seja n um inteiro positivo. Então,*

$$\binom{2n}{n} = \prod_{p \in \mathcal{P}(2n)} p^{k_p},$$

onde $\mathcal{P}(2n)$ é o conjunto dos primos menores ou iguais a $2n$, e

$$k_p = \sum_{m=1}^{\infty} \left(\left\lfloor \frac{2n}{p^m} \right\rfloor - 2 \left\lfloor \frac{n}{p^m} \right\rfloor \right).$$

Além disso, $k_p \leq \lfloor \log_2(2n)/\log_2 p \rfloor$.

Antes de provar o lema, observe que, apesar da soma que define k_p incluir um número infinito de termos, apenas uma quantidade finita destes termos é não nula. De fato, se $p^m > 2n$, então

$$\left\lfloor \frac{2n}{p^m} \right\rfloor = \left\lfloor \frac{n}{p^m} \right\rfloor = 0.$$

DEMONSTRAÇÃO. Vimos na seção anterior que se $k > 0$ é um número inteiro, então existem $\lfloor n/k \rfloor$ múltiplos positivos de k menores que n. Usaremos isto para determinar a multiplicidade de um primo $p \leq n$ na fatoração de $n!$. Pelo argumento acima existem, entre 1 e n,

$$\begin{array}{rl} \lfloor n/p \rfloor & \text{múltiplos de } \ p \\ \lfloor n/p^2 \rfloor & \text{múltiplos de } \ p^2 \\ \lfloor n/p^3 \rfloor & \text{múltiplos de } \ p^3 \\ & \vdots \\ \lfloor n/p^m \rfloor & \text{múltiplos de } \ p^m. \end{array}$$

Portanto, p deve ter multiplicidade

$$\mu(n,p) = \sum_{k=1}^{\infty} \lfloor n/p^k \rfloor.$$

na fatoração de $n!$. Para ver que isto está correto, suponha que $a = p^m b \leq n$ e que p não divida b. Mas isto significa que a contribui com 1 à parcela $\lfloor n/p^k \rfloor$ da soma $\mu(n,p)$, para cada $1 \leq k \leq m$. Portanto, a contribuição total de a à soma $\mu(n,p)$ é exatamente m, como esperado. Como já observamos anteriormente, a soma que define $\mu(n,p)$ é finita, pois $\lfloor n/p^r \rfloor = 0$ se $p^r > n$.

Por outro lado,

$$\binom{2n}{n} = \frac{(2n)!}{n!n!}.$$

Assim, procedendo ao cancelamento dos termos comuns ao numerador e denominador, verificamos que um primo $p \leq 2n$ tem multiplicidade

$$\mu(2n,p) - 2\mu(n,p) \tag{1}$$

na fatoração de $\binom{2n}{n}$. Substituindo em (1) as fórmulas para $\mu(2n,p)$ e $\mu(n,p)$ obtidas acima chegamos ao resultado do lema.

Resta-nos provar a desigualdade final. Suponhamos que p^s é a maior potência de p que divide $2n$. Neste caso, a soma que define k_p tem, no máximo s parcelas, pois $\lfloor 2n/p^t \rfloor = 0$ para $t > s$. Por outro lado, se $a = 2n/p^m$ para algum $m < s$, então

$$0 \leq \left\lfloor \frac{2n}{p^m} \right\rfloor - 2\left\lfloor \frac{n}{p^m} \right\rfloor = \lfloor a \rfloor - 2\lfloor \frac{a}{2} \rfloor < a - 2(\frac{a}{2} - 1) = 2.$$

Deste modo, cada parcela da soma k_p contribui, no máximo, 1 à soma total. Com isto k_p não pode ser maior que s. Finalmente, de $p^s \leq 2n$ segue que $s \log_2 p \leq \log_2(2n)$, donde

$$k_p \leq s \leq \lfloor \log_2(2n)/\log_2 p \rfloor.$$

Seja $n > 0$ um número inteiro. Definimos o *primorial* de n, denotado por $n^\sharp$, como o produto de todos os números primos positivos menores ou iguais a n. Por exemplo,

$$12^\sharp = 2 \cdot 3 \cdot 5 \cdot 7 \cdot 11.$$

O principal teorema desta seção nos dá uma desigualdade envolvendo um primorial.

TEOREMA 1.2. *Se n é um inteiro positivo, então $2^n \leq (2n^2)^\sharp$.*

DEMONSTRAÇÃO. Seja m um inteiro positivo. Pelo lema 1.1, temos que

$$\binom{2m}{m} = \prod_{p \in \mathcal{P}(2m)} p^{k_p},$$

onde $k_p \leq \lfloor \log_2 2m/\log_2 p \rfloor$. Como $p \geq 2$, temos que $k_p \leq \lfloor \log_2 2m \rfloor$. Tomando $r = \lfloor \log_2 2m \rfloor$, concluímos que

$$\binom{2m}{m} \text{ divide } \prod_{p \in \mathcal{P}(2m)} p^r = ((2m)^\sharp)^r.$$

Logo,

$$\binom{2m}{m} \leq ((2m)^\sharp)^r.$$

Mas

$$\binom{2m}{m} = \frac{2m(2m-1)\cdots(m+1)}{m!} = \frac{2m}{m}\frac{(2m-1)}{m-1}\cdots\frac{(m+2)}{2}\frac{(m+1)}{1} \geq 2^m,$$

de modo que

$$2^m \leq ((2m)^\sharp)^r.$$

Seja, agora, $m = n^2$. Então,

$$2^{n^2} \leq ((2n^2)^\sharp)^r$$

Tomando logaritmos na base 2, verificamos que

$$n^2 \leq (\log_2 2n^2) \log_2(((2n^2)^\sharp)),$$

já que $r = \lfloor \log_2 2n^2 \rfloor$. Ou seja,

$$n \leq 2\left(\frac{\log_2(n\sqrt{2})}{n}\right)\log_2(((2n^2)^\sharp)).$$

Como $2\log_2\sqrt{2}n/n \leq 1$ para $n \geq 4$, então

$$n \leq \log_2((2n^2)^\sharp).$$

Portanto,

$$2^n \leq (2n^2)^\sharp,$$

para todo $n \geq 4$. Uma verificação direta mostra que a desigualdade vale também quando n é 2 ou 3.

A demonstração deste teorema é uma adaptação da prova do teorema de Chebyshev, enunciado na introdução. Veja também **[Hardy e Wright 1994, seção 22.2]** ou **[Landau 2002, capítulo 8]**. Uma outra demonstração bastante simples do teorema de Chebyshev pode ser encontrada em **[Nair 1982]**. Para uma resenha dos fatos básicos da teoria analítica de números consulte **[Voloch 1987]**. Uma exposição mais completa está disponível no clássico **[Ingham 1932]**.

5. Exercícios

1. Considere o conjunto de inteiros

$$S = \{4k + 1 : k \in \mathbb{N}\}.$$

Dizemos que $q \in S$ é um S-*primo* se $q = ab$, com $a, b \in S$, implica que $a = 1$ ou $b = 1$.

(a) Mostre que o produto de dois elementos de S também pertence a S.

(b) Mostre que todo elemento de S pode ser fatorado como um produto de S-primos.

(c) Dê exemplos de elementos de S que podem ser decompostos de *duas maneiras diferentes* como produtos de S-primos.

(d) Adapte o Crivo de Eratóstenes de modo a determinar os S-primos menores ou iguais a um elemento $s \in S$.

2. O objetivo deste exercício é dar uma demonstração da infinidade dos números primos proposta por L. Euler em 1737. Considere o produto

$$P = \left(\frac{1}{1-1/2}\right)\left(\frac{1}{1-1/3}\right)\left(\frac{1}{1-1/5}\right)\cdots$$

onde temos um termo para cada número primo.

(a) Use o Teorema da Fatoração Única para mostrar que este produto é igual à série infinita

$$S = 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \frac{1}{5} + \frac{1}{6} + \cdots$$

(b) Mostre que se o número de primos fosse finito S seria convergente.
(c) Mostre que S é divergente.
(d) Conclua que existem infinitos números primos.

3. Sejam f_1, f_2, g_1 e g_2 funções cujo contradomínio é $\mathbb{R}$ e cujo domínio é $\mathbb{N}$ (ou $\mathbb{R}$). Mostre que, se f_1 é $O(g_1)$ e f_2 é $O(g_2)$, então
(1) $f_1 + f_2$ é $O(\max\{g_1, g_2\})$, e
(2) $f_1 f_2$ é $O(g_1 g_2)$.

O objetivo dos exercícios 4, 5, 6 e 7 é obter uma cota para o custo do algoritmo euclidiano. Uma descrição detalhada deste algoritmo pode ser encontrada em **[Coutinho 2001, capítulo 1]**.

4. Considere a *seqüência de Fibonacci*, definida por
$$F_1 = F_2 = 1 \text{ e } F_{n+1} = F_n + F_{n-1}.$$
Prove, por indução em n que
$$F_n > \left(\frac{(1+\sqrt{5})}{2}\right)^2,$$
para todo $n \geq 3$.

5. Sejam $k \geq 1$ e $a > b > 1$ números inteiros. Mostre que, se o cálculo do máximo divisor comum de a e b pelo algoritmo euclidiano executa k divisões, então
$$a \geq F_{k+2} \text{ e } b \geq F_{k+1}.$$

6. Use os exercícios 4 e 5 para mostrar que o número de divisões necessárias para calcular o máximo divisor comum de dois inteiros $a > b > 1$ pelo algoritmo euclidiano é $O(\log_2 b)$. Este resultado foi provado originalmente por G. Lamé em 1845.

7. Mostre que o tempo de execução do algoritmo euclidiano estendido aplicado a dois inteiros positivos menores que n é $O((\log_2 n)^3)$.
SUGESTÃO: Lembre-se que a divisão e a multiplicação de dois inteiros menores que n têm custo $O((\log_2 n)^2)$.

8. Suponha que você dispõe de um computador que efetua qualquer operação binária em 10^{-9} segundos. Determine o tamanho n do maior problema que pode ser resolvido em 1 segundo por um algoritmo cujo tempo de execução é:
(a) $(\log_2 n)^2$;
(b) $(\log_2 n)^{33}$;
(c) n;
(d) n^2.

9. Mostre que
$$\sum_{k \leq n} 1/k \leq \ln n$$
e use isto para dar uma avaliação do custo do crivo de Eratóstenes.

10. Mostre que, se $n > 2$ é um inteiro par, então $n^\sharp = (n-1)^\sharp$.

11. Seja m um inteiro positivo. Mostre que o número binomial
$$M = \binom{2m+1}{m}.$$
satisfaz $M < 2^{2m}$.

12. Prove que $(2m+1)^\sharp/(m+1)^\sharp$ divide $\binom{2m+1}{m}$ e conclua que
$$\frac{(2m+1)^\sharp}{(m+1)^\sharp} \leq M < 2^{2m}.$$

13. Use os dois exercícios anteriores para provar, por indução em n, que $n^\sharp < 2^n$. Considere separadamente os casos em que n é par e n é ímpar. O primeiro caso é praticamente imediato, o segundo requer um argumento simples envolvendo desigualdades.

Capítulo 2

Grupos abelianos

Neste capítulo introduzimos grupos abelianos, seus subgrupos e quocientes, e provamos o teorema de Lagrange.

1. Definição e exemplos

Os grupos são exemplos de estruturas algébricas; isto é, são conceitos de natureza classificatória, cujo objetivo é semelhante ao das classes taxonômicas em biologia. A principal razão para definir a classe das aves, por exemplo, é tornar possível descrever, com uma palavra, uma série de animais com características comuns. Assim, é provável que você nunca tenha ouvido falar de um kakapo, mas se alguém lhe disser que um kakapo é uma ave, você imediatamente vai imaginá-lo com duas patas, penas e bico [1].

Em álgebra acontece a mesma coisa. Da observação de exemplos com propriedades comuns surge a idéia de definir uma classe (ou estrutura) à qual pertencem estes vários exemplos. A diferença está no fato das estruturas matemáticas serem muito mais simples e rígidas que as da biologia. Assim, uma definição satisfatória, em matemática, permite-nos deduzir uma cascata de novas propriedades que não estavam explicitamente presentes na definição original. Por exemplo, das propriedades básicas dos números inteiros, e da definição de número primo, segue que existe uma infinidade de primos. No entanto, na definição de primo, não há nenhuma indicação quanto à sua quantidade.

Neste livro estaremos utilizando duas estruturas algébricas básicas, os grupos abelianos e os anéis. Começamos por estudar os grupos abelianos. Os anéis ficam para o próximo capítulo.

Um *grupo* é constituído de dois ingredientes básicos: um *conjunto* e uma *operação* definida neste conjunto. Vamos chamar o conjunto de G e a operação de $\star$. Por operação estamos entendendo uma regra que a cada dois elementos $a, b \in G$ associa um terceiro elemento $a \star b$ que também está em G. É importante que você se dê conta, desde já, de que a palavra 'operação' é usada em matemática em um contexto mais geral do que o que estamos adotando neste livro. Por exemplo, costumamos nos referir ao produto interno de vetores como uma operação. Apesar disso ele não se encaixa na definição acima porque, a cada dois vetores, é associado um número, e não um vetor.

[1] Um kakapo é um papagaio neozeolandês que não voa, e que corre sério risco de extinção. Para mais informações (e fotos) visite `www.kakaporecovery.org.nz`

Apesar de muitos conjuntos utilizados em matemática estarem naturalmente associados a operações, nem sempre temos um grupo. Para isso, precisamos que a operação satisfaça algumas propriedades. Um conjunto G, onde está definida uma operação $\star$, é um *grupo* se:

Associatividade: dados $a, b, c \in G$ temos que

$$a \star (b \star c) = (a \star b) \star c.$$

Elemento neutro: existe um elemento $e \in G$ tal que, para todo $a \in G$, temos

$$a \star e = e \star a = a.$$

Elemento inverso: dado um elemento $a \in G$, existe um elemento $a' \in G$ (o **inverso** de a) tal que

$$a \star a' = a' \star a = e.$$

Observe que *não* exigimos que a operação do grupo seja comutativa; isto é, que

$$a \star b = b \star a \text{ para quaisquer elementos } a, b \in G.$$

Um grupo cuja operação é comutativa é chamado de *grupo abeliano*, em homenagem ao matemático norueguês Niels Henrik Abel, que foi um dos pioneiros da teoria de grupos. Apesar dos grupos não abelianos aparecerem com freqüência em matemática e nas suas aplicações (à física, por exemplo), todos os grupos utilizados neste livro são abelianos. Por causa disso, trabalharemos, de agora em diante, apenas com grupos abelianos. Isto significa que, mesmo os resultados que são verdadeiros para grupos em geral, serão provados apenas para grupos cuja operação é comutativa.

Há muitos exemplos importantes da associação 'conjunto e operação' que *não* satisfazem as propriedades acima e, portanto, não são grupos abelianos. Por exemplo, a soma de números naturais é associativa e tem elemento neutro (o zero). Entretanto o único natural a ter inverso para a soma é o zero; porque os números negativos não são naturais. Apesar disto, os grupos podem ser encontrados por toda parte. Assim, o conjunto dos números inteiros, racionais, reais e complexos são grupos abelianos com respeito à adição. Para qualquer inteiro positivo n, o conjunto $\mathbb{Z}_n$ é um grupo relativamente à adição de classes. Já os racionais, reais e complexos *não nulos* são grupos abelianos relativamente à multiplicação.

Grupos abelianos cuja operação é denotada com o símbolo $+$ são conhecidos como *grupos aditivos*. Esta terminologia é especialmente apropriada quando dois grupos convivem em um mesmo conjunto. Por exemplo, $\mathbb{Q}$ é um grupo com respeito à adição; já $\mathbb{Q} \setminus \{0\}$ é um grupo relativamente à multiplicação. Assim, a expressão "o grupo aditivo $\mathbb{Q}$" refere-se ao primeiro destes grupos. No caso da operação de um grupo G ser denotada como uma adição, o 'inverso' de um elemento $a \in G$ é denotado por $-a$, e chamado de *simétrico* de a. Um grupo cuja operação não é denotada por $+$ será chamado de *grupo multiplicativo*, independente do fato de sua operação ser denotada por

justaposição, por um ponto, ou por um símbolo exótico como $\star$. Neste caso, o inverso de um elemento a será escrito a^{-1}.

O número de elementos de um grupo é a sua *ordem*. Todos os grupos mencionados anteriormente têm ordem infinita, exceto o grupo aditivo $\mathbb{Z}_n$, que tem ordem n. Outro grupo abeliano finito fácil de descrever é $\{-1, 1\}$ com a operação de multiplicação de inteiros. Neste caso a ordem é 2.

Não podemos perder de vista que nosso interesse último é o estudo das propriedades de $\mathbb{Z}$, sobretudo as de caráter *multiplicativo*, que estão relacionadas a primos e fatoração. Sob este ponto de vista, um dos grupos abelianos mais importantes é o conjunto $U(n)$, dos elementos invertíveis de $\mathbb{Z}_n$. Em outras palavras,

$$U(n) = \{\overline{a} \in \mathbb{Z}_n : \mathrm{mdc}(a, n) = 1\}.$$

Este conjunto é um grupo abeliano para a operação de multiplicação de classes de $\mathbb{Z}_n$. Para verificar isto lembre-se, em primeiro lugar, que o produto de dois elementos de $\mathbb{Z}_n$ é um elemento de $\mathbb{Z}_n$. Portanto, se multiplicarmos dois elementos de $U(n)$ obteremos um elemento de $\mathbb{Z}_n$. O problema é que isto não basta: precisamos que o produto de duas classes que estejam em $U(n)$ também pertença a $U(n)$. Em outras palavras, devemos provar que o produto de dois elementos invertíveis $\overline{a}$ e $\overline{b}$ de $\mathbb{Z}_n$ também é invertível. Digamos que $\overline{a'}$ e $\overline{b'}$ são os inversos de $\overline{a}$ e $\overline{b}$, respectivamente. Então,

$$\overline{ab} \cdot \overline{a'b'} = \overline{aa'} \cdot \overline{bb'} = \overline{1},$$

de modo que $\overline{ab}$ tem inverso $\overline{a'b'}$ em $\mathbb{Z}_n$. Logo, $\overline{ab} \in U(n)$, e a operação de multiplicação de classes está bem definida em $U(n)$.

Resta-nos verificar que esta operação satisfaz as propriedades requeridas. A associatividade e a comutatividade são fáceis de comprovar, porque sabemos que o produto em $\mathbb{Z}_n$ é associativo e comutativo. O elemento neutro é $\overline{1}$, que é invertível em $\mathbb{Z}_n$ e, portanto, está em $U(n)$. Finalmente, a definição de $U(n)$ requer que cada um de seus elementos tenha inverso multiplicativo. Portanto, $U(n)$ é um *grupo abeliano* relativamente ao produto de classes.

É fácil dar uma descrição completa de um grupo finito G, de ordem pequena, através da *tabela de multiplicação* deste grupo. Primeiramente, cada linha e coluna da tabela são indexadas por um elemento do grupo. Se $x, y \in G$, então a casa da tabela que fica na intersecção da linha indexada por x com a coluna indexada por y é preenchida com $x \star y$. Para o grupo $U(20)$ a tabela aparece na página seguinte.

Observe que quando nos referimos às casas de uma linha estamos sempre excluindo a que está mais à esquerda, que serve apenas para indexar a linha na tabela. O mesmo vale para a casa no topo de cada coluna. Levando isto em conta, podemos relacionar várias propriedades de um grupo a características de sua tabela. Por exemplo:

- os elementos de uma mesma linha de um grupo são todos distintos,
- os elementos de uma mesma coluna de um grupo são todos distintos,
- um grupo é abeliano se sua tabela for simétrica em relação à diagonal.

$\cdot$	$\overline{1}$	$\overline{3}$	$\overline{7}$	$\overline{9}$	$\overline{11}$	$\overline{13}$	$\overline{17}$	$\overline{19}$
$\overline{1}$	$\overline{1}$	$\overline{3}$	$\overline{7}$	$\overline{9}$	$\overline{11}$	$\overline{13}$	$\overline{17}$	$\overline{19}$
$\overline{3}$	$\overline{3}$	$\overline{9}$	$\overline{1}$	$\overline{7}$	$\overline{13}$	$\overline{19}$	$\overline{11}$	$\overline{17}$
$\overline{7}$	$\overline{7}$	$\overline{1}$	$\overline{9}$	$\overline{3}$	$\overline{17}$	$\overline{11}$	$\overline{19}$	$\overline{13}$
$\overline{9}$	$\overline{9}$	$\overline{7}$	$\overline{3}$	$\overline{1}$	$\overline{19}$	$\overline{17}$	$\overline{13}$	$\overline{11}$
$\overline{11}$	$\overline{11}$	$\overline{13}$	$\overline{17}$	$\overline{19}$	$\overline{1}$	$\overline{3}$	$\overline{7}$	$\overline{9}$
$\overline{13}$	$\overline{13}$	$\overline{19}$	$\overline{11}$	$\overline{17}$	$\overline{3}$	$\overline{9}$	$\overline{1}$	$\overline{7}$
$\overline{17}$	$\overline{17}$	$\overline{11}$	$\overline{19}$	$\overline{13}$	$\overline{7}$	$\overline{1}$	$\overline{9}$	$\overline{3}$
$\overline{19}$	$\overline{19}$	$\overline{17}$	$\overline{13}$	$\overline{11}$	$\overline{9}$	$\overline{7}$	$\overline{3}$	$\overline{1}$

TABELA 1. Tabela de $U(20)$

Naturalmente a diagonal da tabela é formada pelas casas que ficam no cruzamento de uma linha e uma coluna indexadas pelo mesmo elemento do grupo. Estas propriedades são fáceis de provar, e os detalhes ficam por sua conta.

2. Subgrupos

Se um conjunto que está dentro de outro conjunto é chamado de subconjunto, então parece justo chamar um grupo que está contido em outro de subgrupo. Entretanto, precisamos tomar um pouco de cuidado com esta definição. Por exemplo, $\mathbb{Q} \setminus \{0\}$ é um grupo para a multiplicação e está contido no grupo aditivo $\mathbb{Q}$. Porém, como as operações nestes dois grupos são completamente diferentes, um não herda as propriedades do outro. Mas isto é extremamente inconveniente. Uma das vantagens de saber que H é subgrupo de G deveria ser que podemos descobrir algumas propriedades de H apenas do fato de que está contido em G. Para tornar isto viável, precisamos não apenas que $H \subseteq G$, e que ambos sejam grupos, mas também que *eles compartilhem a mesma operação*. Para sermos mais precisos, daremos uma definição formal.

Seja G um grupo abeliano com a operação $\star$. Um subconjunto não vazio H de G é um *subgrupo* de G se:

(i) Para todo $a, b \in H$ temos que $a \star b \in H$.
(ii) O elemento neutro de G está em H.
(iii) Para todo $a \in H$, seu inverso a^{-1} também está em H.

Observe que (i) apenas afirma que a operação $\star$ de G também é uma operação para H. Talvez você estranhe o fato de que, apesar de nossa discussão anterior sugerir que H deveria ser um grupo relativamente a $\star$, a propriedade associativa não foi listada entre as condições que H deve satisfazer. A razão disto é que, como G é um grupo, então a associatividade de $\star$ como operação

de G nos garante que

$$a \star (b \star c) = (a \star b) \star c,$$

quaisquer que sejam $a, b, c \in G$. Em particular, como $H \subseteq G$, então esta propriedade tem que ser verificada se escolhermos $a, b, c \in H$. Mas isto significa que $\star$ é automaticamente associativa como operação em H. Um argumento semelhante se aplica à comutatividade, de modo que se G for um grupo abeliano, o mesmo vai ocorrer com H.

Entre os exemplos listados na seção 1, há vários subgrupos. Por exemplo, $\mathbb{Z}$ é um subgrupo aditivo de $\mathbb{Q}$, que é um subgrupo aditivo de $\mathbb{R}$, que é um subgrupo aditivo de $\mathbb{C}$. No caso da operação ser a multiplicação: os racionais não nulos são um subgrupo dos reais não nulos, que por sua vez são um subgrupo dos complexos não nulos.

Um exemplo importante é o dos subgrupos do grupo aditivo $\mathbb{Z}$. Seja n um inteiro positivo, e considere o conjunto H_n dos múltiplos de n; isto é,

$$H_n = \{kn : k \in \mathbb{Z}\}.$$

As propriedades básicas do cálculo com múltiplos de n garantem que H_n é um subgrupo de $\mathbb{Z}$. Assim, a condição (i), da definição de subgrupo, é equivalente à soma dos múltiplos de n ser um múltiplo de n; a condição (ii) corresponde a 0 ser múltiplo de n, e (iii) segue do fato de que a é múltiplo de n se, e somente se, $-a$ também é. Pode-se mostrar que qualquer subgrupo do grupo aditivo $\mathbb{Z}$ é igual a H_n para algum n; veja exercício 1.

Dado um grupo G qualquer, é evidente que o maior subgrupo de G é o próprio G. Mas qual será o menor subgrupo de G? De acordo com a condição (ii), este subgrupo tem que conter o elemento neutro e. Contudo, o conjunto $\{e\}$, formado apenas pelo elemento neutro, também satisfaz (i) e (iii). Logo, $\{e\}$ é um subgrupo de G. Portanto, seja qual for o grupo G, ele contém pelo menos dois subgrupos: $\{e\}$, que é o menor dos subgrupos e *está contido* em todos os outros, e G, que é o maior, e *contém* todos os outros subgrupos.

Por causa das aplicações que temos em vista, estamos mais interessados nos grupos finitos. Por isso, vamos nos restringir a este tipo de grupos de agora em diante.

Seja G um grupo abeliano *finito* provido de uma operação $\star$, e cujo elemento neutro é e. Digamos que $a \neq e$ é um elemento de G. Seja $k \geq 0$ um inteiro. Vamos escrever

$$a^k = \begin{cases} \underbrace{a \star a \star \cdots \star a}_{k \text{ vezes}}, & \text{se } k > 0 \\ e & \text{se } k = 0 \end{cases}$$

Esta é a k-ésima *potência* de a. Consideremos o conjunto das potências de a

$$H = \{e, a, a^2, a^3, \ldots\}.$$

Olhando para a seqüência de expoentes podemos ter a impressão de que este conjunto é infinito. Contudo, como G é um grupo, as potências de a estão

contidas em G. Logo, $H \subseteq G$. Mas G é um conjunto finito; portanto, H tem que ser finito também.

Como temos um número infinito de expoentes correspondendo a um número finito de elementos, o *Princípio da Casa do Pombo* nos diz que a um mesmo elemento de H corresponderão expoentes distintos. Isto é, existem inteiros positivos $n > m$, tais que $a^m = a^n$. Seja a^{-1} o inverso de a em G. Multiplicando $a^m = a^n$ por $(a^{-1})^m$, obtemos $a^{n-m} = e$. Como $n - m > 0$, concluímos que, dado um elemento qualquer $a \in G$, existe um inteiro positivo ℓ tal que $a^\ell = e$.

Digamos que k é o menor inteiro positivo tal que $a^k = e$. Se $n > k$, então podemos dividir n por k, obtendo $n = kq + r$, onde $0 \leq r \leq k - 1$. Então,

$$a^n = a^{kq+r} = (a^k)^q \star a^r.$$

Como $a^k = e$, temos que $a^n = a^r$. Por isso, não precisamos nos preocupar com as potências de a com expoente maior que $k - 1$, porque são iguais a outras potências, de expoentes menores. Assim,

$$H = \{e, a, a^2, \ldots, a^{k-1}\}, \tag{2}$$

que são todos elementos distintos, como é fácil verificar; veja exercício 2. Em particular, H tem k elementos.

Mas será que H é um subgrupo de G? Para começar, segue imediatamente da definição de potência em G, que

$$a^r \star a^s = a^{r+s}.$$

Portanto, o produto $x \star y$ de dois elementos de $x, y \in H$ é um elemento de H. Como $e \in H$, por definição, precisamos apenas mostrar que se $r > 0$, então a^r admite como inverso uma outra potência de a cujo expoente é positivo. A descrição de H em (2) nos permite assumir, sem perda de generalidade, que $r < k$. Neste caso, $k - r > 0$ e

$$a^r \star a^{k-r} = a^k = e.$$

Logo, a^{k-r} é o inverso de a^r em H. Mostramos, portanto, que H é um subgrupo de G, cuja ordem é k.

Podemos adotar esta construção como um procedimento para produzir subgrupos de G. Mais precisamente, dado um grupo finito G, e um elemento $a \in G$, temos que:

(i) O conjunto H formado pelas potências de a em G é um subgrupo, que será denotado por $\langle a \rangle$.

(ii) A ordem de $\langle a \rangle$ é igual ao menor inteiro positivo k tal que $a^k = e$.

É conveniente introduzir a seguinte terminologia. Se um subgrupo H é igual ao conjunto das potências de um elemento a, diremos que a é um *gerador* de $H = \langle a \rangle$. O menor inteiro positivo k tal que $a^k = e$ é a *ordem* de a. Portanto, se H tem a como gerador então a ordem de $H = \langle a \rangle$ (isto é, o número de elementos de H) é igual à ordem de a. Um subgrupo que admite um elemento

gerador é chamado de *cíclico*. Uma conseqüência imediata da definição é que um subgrupo de ordem k só será cíclico se contiver um elemento de ordem k.

Evidentemente todo grupo contém subgrupos cíclicos. Contudo, nem todos os subgrupos de um grupo têm que ser cíclicos. Por exemplo, no grupo

$$U(8) = \{\overline{1}, \overline{3}, \overline{5}, \overline{7}\}$$

das classes invertíveis módulo 8, cada elemento tem ordem 2. Portanto, o grupo $U(8)$ não pode ser cíclico, já que sua ordem é 4.

Para incluir o que acontece com $U(8)$, podemos generalizar nossa discussão anterior a grupos que são gerados por vários elementos. Seja G um grupo abeliano finito e digamos que $a_1, \ldots, a_t$ são elementos de G. Considere o conjunto

$$\{a_1^{r_1} \star \cdots \star a_t^{r_t} : r_1, \ldots r_t \text{ são inteiros não negativos } \},$$

que vamos denotar por $\langle a_1, \ldots, a_t \rangle$. Se a_j tiver ordem k_j em G, então podemos supor, sem perda de generalidade, que $0 \leq r_j \leq k_j$ para $1 \leq j \leq t$. Em particular, $\langle a_1, \ldots, a_t \rangle$ tem, no máximo $k_1 k_2 \cdots k_t$ elementos. É evidente que o produto de dois elementos de $\langle a_1, \ldots, a_t \rangle$ está contido em $\langle a_1, \ldots, a_t \rangle$. Como o elemento neutro é obtido tomando todos os expoentes dos as iguais a 0, temos que $e \in \langle a_1, \ldots, a_t \rangle$. Portanto, para confirmar que este conjunto é um subgrupo de G basta verificar que cada um dos seus elementos tem inverso. Deixaremos os detalhes desta parte do argumento para você completar; veja exercício 3.

Diremos que $\langle a_1, \ldots, a_t \rangle$ é o *subgrupo de G gerado pelos elementos* $a_1, \ldots, a_t$. Os subgrupos cíclicos são um caso particular desta construção quando $t = 1$. Aplicando isto a $U(8)$ verificamos que, embora não seja um grupo cíclico, pode ser gerado por quaisquer dois de seus elementos que sejam diferentes de $\overline{1}$.

Determinar os subgrupos de um grupo dado não é uma tarefa fácil. Contudo, como veremos na próxima seção, existem relações entre a ordem de um grupo, e as ordens de seus subgrupos e elementos, que facilitam um pouco esta tarefa.

3. Teorema de Lagrange

Curiosamente, o fato de um conjunto finito G admitir uma operação que satisfaz as propriedades de um grupo faz com que existam muitas relações numerológicas curiosas entre o número de elementos de G e o número de elementos de seus subgrupos. Isto torna mais fácil a tarefa de determinar os subgrupos de um dado grupo finito. A mais simples, e provavelmente a mais útil, destas relações, é dada pelo *teorema de Lagrange*.

TEOREMA DE LAGRANGE. *Em um grupo finito, a ordem de qualquer subgrupo divide a ordem do grupo.*

Como conseqüência disto, a ordem de qualquer elemento a de um grupo finito G também divide a ordem de G. Afinal, pelo teorema de Lagrange, a ordem do subgrupo cíclico $\langle a \rangle$ gerado por a divide a ordem de G. Mas, como vimos na seção anterior, a ordem de $\langle a \rangle$ é igual à ordem de a. Logo, a ordem de a divide a ordem de G.

A demonstração do teorema de Lagrange será feita na seção 4. Antes precisamos nos convencer da utilidade deste resultado na determinação dos subgrupos de um dado grupo. Começaremos com o grupo $U(20)$, cuja tabela descrevemos na seção 1. Este grupo tem ordem 8. Logo, pelo teorema de Lagrange, só pode ter subgrupos de ordem 1, 2, 4 ou 8. Já conhecemos os subgrupos de ordens 1 e 8, que são $\{\overline{1}\}$ e $U(20)$, respectivamente.

Vamos determinar as ordens dos elementos de $U(20)$. Um cálculo fácil mostra que os elementos de ordem 2 são $\overline{9}$, $\overline{11}$, $\overline{19}$ e os elementos de ordem 4 são $\overline{3}$, $\overline{7}$, $\overline{13}$ e $\overline{17}$. Nossa primeira conclusão é que $U(20)$ não é cíclico porque não tem nenhum elemento de ordem 8. Os subgrupos cíclicos de ordem 2 de $U(20)$ são

$$\langle \overline{9} \rangle, \langle \overline{11} \rangle \text{ e } \langle \overline{19} \rangle.$$

e os de ordem 4 são

$$\langle \overline{3} \rangle = \langle \overline{7} \rangle \text{ e } \langle \overline{13} \rangle = \langle \overline{17} \rangle.$$

Observe que um subgrupo qualquer de ordem 2, por ter apenas dois elementos, tem que ser cíclico. E quanto aos subgrupos de ordem 4? Suponhamos que H seja um subgrupo de ordem 4 de $U(20)$, *que não é cíclico*. Neste caso, H não pode ter nenhum elemento de ordem 4. Como a ordem de um elemento de H tem que dividir a ordem de H, concluímos que os elementos de H diferentes de $\overline{1}$ têm que ter ordem 2. Portanto, precisamos de três elementos de ordem 2 em $U(20)$ para ter chance de obter um subgrupo não cíclico de ordem 4. Por sorte, $U(20)$ tem exatamente três elementos de ordem 2, e uma verificação rápida mostra que o conjunto

$$H = \{\overline{1}, \overline{9}, \overline{11}, \overline{19}\}.$$

de fato é um subgrupo de $U(20)$.

O fato de $U(20)$ ter exatamente 3 elementos de ordem 2 pode deixar uma impressão errada quanto à real dificuldade em determinar subgrupos de ordem 2. Para dirimir qualquer dúvida, faremos outro exemplo. Desta vez o grupo é $U(40)$, que tem 16 elementos. Já sabemos que, para determinar os subgrupos não cíclicos de ordem 4 precisamos apenas conhecer os elementos de ordem 2 de $U(40)$, que são $\overline{9}, \overline{11}, \overline{19}, \overline{21}, \overline{29}, \overline{31}$ e $\overline{39}$. Como temos 7 destes elementos, existem

$$\binom{7}{3} = 35$$

maneiras de agrupá-los em subconjuntos de 3 elementos. Entretanto, não basta acrescentar $\overline{1}$ a cada um destes subconjuntos para obter um subgrupo de $U(40)$. Afinal, ao multiplicar dois elementos quaisquer de um subgrupo, precisamos

obter um outro elemento que também está no subgrupo. Isto significa, por exemplo, que se H é um subgrupo de $U(40)$ que contém $\overline{9}$ e $\overline{11}$, então

$$\overline{9} \cdot \overline{11} = \overline{19}, \tag{3}$$

também tem que estar em H. Como estamos supondo que H tem ordem 4, obtemos

$$H = \{\overline{1}, \overline{9}, \overline{11}, \overline{19}\}.$$

Um cálculo fácil mostra que todos os outros produtos possíveis entre elementos de H pertencem a H, o que comprova que H é um subgrupo de $U(40)$. Por outro lado,

$$\{\overline{1}, \overline{9}, \overline{11}, \overline{21}\},$$

não é um subgrupo de $U(40)$, porque por (3) o produto $\overline{9} \cdot \overline{11}$ não pertence a este conjunto.

Uma outra aplicação simples do teorema de Lagrange nos permite descrever a estrutura de um grupo G cuja ordem é um número primo p. Escolha um elemento qualquer $a \neq e$ de G e considere o subgrupo cíclico $\langle a \rangle$ gerado por a. Pelo teorema de Lagrange, a ordem de $\langle a \rangle$ tem que dividir a ordem de G, que é p, um número primo. Logo $\langle a \rangle$ tem ordem 1 ou p. Mas $\langle a \rangle$ contém pelo menos dois elementos, a saber e e a. Logo $\langle a \rangle$ não pode ter ordem 1; portanto, deve ter ordem p. Como $\langle a \rangle$ tem tantos elementos quantos tem G, concluímos que $\langle a \rangle = G$. Para referência futura, reuniremos isto em um teorema.

TEOREMA 2.1. *Todo grupo G de ordem prima é cíclico e qualquer elemento de G, diferente de e, é um gerador de G.*

É preciso um certo cuidado na interpretação deste resultado, uma vez que não estamos afirmando que todo grupo cíclico tem ordem prima. Na verdade, isto é falso. Por exemplo,

$$U(5) = \{\overline{1}, \overline{2}, \overline{3}, \overline{4}\}$$

tem ordem 4, um número composto, mas pode ser gerado por $\overline{2}$. Contudo, $\overline{4}$ não é um gerador de $U(5)$. Sobre isto veja também os exercícios 4 e 5.

Antes de encerrar a seção, há um outro resultado que precisamos considerar. Apesar de não ser conseqüência do teorema de Lagrange, seu carácter numerológico faz com que se encaixe, de maneira natural, nesta seção.

LEMA CHAVE. *Digamos que G é um grupo finito munido de uma operação $\star$. Seja $a \in G$. Um inteiro positivo t satisfaz $a^t = e$ se, e somente se, t é divisível pela ordem de a.*

A demonstração é simples. Vamos chamar de s a ordem de a. Se s divide t então $t = sr$, para algum inteiro positivo r, e

$$a^t = (a^s)^r = e.$$

A recíproca é menos imediata. Suponhamos que $a^t = e$. Como a ordem é o *menor* inteiro positivo s tal que $a^s = e$, então $s \leq t$. Dividindo t por s, temos

$$t = sq + r \quad \text{onde} \quad 0 \leq r < s.$$

Assim,

$$e = a^t = (a^s)^q \star a^r = a^r,$$

já que $a^s = e$. Como $r < s$ isto só pode acontecer se $r = 0$, ou teríamos uma contradição com o fato de s ser a ordem de a.

4. Grupos quocientes

Nosso objetivo nesta seção é construir um novo grupo a partir de um grupo dado e de um de seus subgrupos. A construção é uma generalização daquela usada para a aritmética modular. Como no caso de $\mathbb{Z}_n$, começaremos definindo uma relação de equivalência.

Seja G um grupo *abeliano* provido de uma operação $\star$, e seja H um subgrupo de G. Definiremos uma relação de equivalência em G usando H. Sejam x e y elementos de G. Dizemos que x e y são *congruentes módulo* H, e escrevemos $x \equiv y \pmod{H}$ se

$$x \star y^{-1} \in H.$$

Lembre-se que y^{-1} é o inverso de y em G.

Vejamos porque esta construção generaliza a congruência módulo n. Neste caso, o grupo G será o conjunto $\mathbb{Z}$ com a operação de soma, e o subgrupo H será o conjunto H_n dos múltiplos de um inteiro positivo n. Vimos na seção 2 que este é um subgrupo de $\mathbb{Z}$. Sejam x, y inteiros. Como o inverso de y no grupo aditivo $\mathbb{Z}$ é $-y$, temos que

$$x \equiv y \pmod{H_n} \text{ se, e somente se, } x - y \in H_n.$$

Isto é, $x - y$ tem que ser um múltiplo de n. Mas isto significa que $x \equiv y \pmod{H_n}$ é a mesma coisa que $x \equiv y \pmod{n}$.

Voltando ao caso geral, em que G é um grupo abeliano e H é um subgrupo de G, precisamos verificar que a relação de congruência módulo H é de equivalência. Faremos isto testando cada uma das propriedades que uma relação de equivalência precisa satisfazer. Suponhamos que $x, y, z \in G$. Em primeiro lugar, como $x \star x^{-1} = e$, o elemento neutro, e $e \in H$ então $x \equiv x \pmod{H}$. Portanto, a relação é reflexiva. Para mostrar que é simétrica, suponha que $x \equiv y \pmod{H}$. Neste caso, $x \star y^{-1} \in H$. Mas um subgrupo contém o inverso de cada um de seus elementos. Logo $y \star x^{-1} = (x \star y^{-1})^{-1}$, pertence a H. Contudo $y \star x^{-1} \in H$ significa que $y \equiv x \pmod{H}$. Finalmente, digamos que $x \equiv y \pmod{H}$ e $y \equiv z \pmod{H}$. Estas duas equações se traduzem como $x \star y^{-1} \in H$ e $y \star z^{-1} \in H$. Como o produto de dois elementos de H está em H, temos que $(x \star y^{-1}) \star (y \star z^{-1}) \in H$. Efetuando o cálculo obtemos $x \star z^{-1} \in H$; que é equivalente a dizer que $x \equiv z \pmod{H}$. Em outras palavras, a congruência módulo H também é transitiva.

Observe que cada uma das condições necessárias para que H seja um subgrupo corresponde, exatamente, a uma das propriedades que a congruência módulo H tem que satisfazer para ser uma relação de equivalência.

Agora que sabemos que a congruência módulo H é uma relação de equivalência, determinaremos a classe de equivalência $\widehat{x}$ de um elemento $x \in G$ para esta relação. Por definição, a classe de equivalência de x é

$$\widehat{x} = \{y \in G : y \equiv x \pmod{H}\}.$$

Mas $y \equiv x \pmod{H}$ é o mesmo que dizer que $y \star x^{-1} = h \in H$. Isto é, $y = h \star x$. Portanto, a classe de equivalência de x é o conjunto

$$\widehat{x} = \{h \star x : h \in H\}. \tag{4}$$

Segue imediatamente da definição que a igualdade $\widehat{x} = \widehat{y}$ é verdadeira se, e somente se, $x \equiv y \pmod{H}$. Em particular, a classe do elemento neutro e é igual a H; isto é, $x \in H$ se, e somente se $\widehat{x} = \widehat{e}$. O conjunto das classes de equivalência de G pela congruência módulo H será denotado por G/H.

Vejamos um exemplo. No grupo $U(20)$, considere o subgrupo

$$H = \{\overline{1}, \overline{9}, \overline{11}, \overline{19}\}.$$

Como escrever $\widehat{\overline{a}}$ deixa a notação muito sobrecarregada, denotaremos a classe de $\overline{a}$ para a congruência módulo H simplesmente por $\widehat{a}$. Adotaremos esta convenção de simplificar a notação sempre que estivermos trabalhando com quocientes de subgrupos de $U(n)$.

Levando em conta esta convenção temos, do que já foi dito acima, que $\widehat{1} = H$. Como $U(20)$ tem 8 elementos, verificamos que há 4 elementos fora de $\widehat{1}$. Portanto, deve haver pelo menos mais uma classe de equivalência para a congruência módulo H. Como $\overline{3} \notin \widehat{1}$, podemos começar calculando $\widehat{3}$. Temos, por definição, que

$$\widehat{3} = \{\overline{3} \cdot \overline{1},\ \overline{3} \cdot \overline{9},\ \overline{3} \cdot \overline{11},\ \overline{3} \cdot \overline{19}\} = \{\overline{3}, \overline{7}, \overline{13}, \overline{17}\}.$$

Como $\widehat{3}$ contém todos os elementos de $U(20)$ que não estão em $\widehat{1}$, então

$$G/H = \{\widehat{1}, \widehat{3}\}.$$

Se escolhermos outro subgrupo relativo ao qual calcular a congruência, obteremos um conjunto de classes totalmente diferente. Por exemplo, tomando H como sendo o grupo de ordem 2 gerado por $\overline{9}$, obtemos as seguintes classes de equivalência

$$\begin{aligned}
\widehat{1} &= \{\overline{1}, \overline{9}\} \\
\widehat{3} &= \{\overline{3}, \overline{7}\} \\
\widehat{11} &= \{\overline{11}, \overline{19}\} \\
\widehat{13} &= \{\overline{13}, \overline{17}\}.
\end{aligned}$$

Por enquanto as classes nos contemplam, estáticas. Mas podemos torná-las mais interessantes, introduzindo uma operação no conjunto das classes. Voltemos, mais uma vez, ao caso geral em que G é um grupo abeliano e H é um subgrupo de G. Sejam a e b elementos de G. Queremos definir uma

operação no conjunto G/H a partir da operação $\star$ do grupo G. Tendo por inspiração a definição da soma de classes em $\mathbb{Z}_n$, podemos escrever

$$\widehat{a} \star \widehat{b} = \widehat{a \star b}. \tag{5}$$

De uma maneira geral, a operação de G/H será denotada pelo mesmo símbolo usado para a operação de G.

Antes de mais nada precisamos verificar que esta operação está bem definida. O ponto é que queremos multiplicar *classes*, mas estamos definindo isto a partir da multiplicação de *representantes* destas classes em G. Acontece que uma mesma classe pode ser representada por vários elementos distintos de G. Portanto, precisamos mostrar que, mesmo efetuando a multiplicação de classes a partir de representantes diferentes, obtemos sempre a mesma classe.

Mais precisamente, suponha que a, α, b e β são elementos de G, e que

$$\widehat{a} = \widehat{\alpha} \text{ e } \widehat{b} = \widehat{\beta}.$$

Queremos mostrar que $\widehat{a \star b} = \widehat{\alpha \star \beta}$. Mas, pelas igualdades de classes acima, temos que

$$a \star \alpha^{-1} \in H \quad \text{e} \quad b \star \beta^{-1} \in H.$$

Multiplicando estes elementos de H, e levando em conta que H é um subgrupo do grupo *abeliano* G, concluímos que

$$(a \star \alpha^{-1}) \star (b \star \beta^{-1}) = (a \star b) \star (\alpha \star \beta)^{-1} \in H.$$

Isto confirma que a operação $\star$ de G/H, dada pela equação (5), está bem definida. Observe que só pudemos escrever a última igualdade acima porque sabemos que o grupo é abeliano.

Voltando a um exemplo já considerado anteriormente, seja $H = \{\overline{1}, \overline{9}\}$ o subgrupo de ordem 2 de $U(20)$ cujas classes calculamos acima. Como, $\overline{3} \cdot \overline{11} = \overline{13}$, então

$$\widehat{3} \cdot \widehat{11} = \widehat{13}$$

Como $U(20)/H$ tem apenas 4 elementos, podemos tabelar todas as possibilidades de multiplicação de suas classes, à maneira da seção 1.

$\cdot$	$\widehat{1}$	$\widehat{3}$	$\widehat{11}$	$\widehat{13}$
$\widehat{1}$	$\widehat{1}$	$\widehat{3}$	$\widehat{11}$	$\widehat{13}$
$\widehat{3}$	$\widehat{3}$	$\widehat{1}$	$\widehat{13}$	$\widehat{11}$
$\widehat{11}$	$\widehat{11}$	$\widehat{13}$	$\widehat{1}$	$\widehat{3}$
$\widehat{13}$	$\widehat{13}$	$\widehat{11}$	$\widehat{3}$	$\widehat{1}$

TABELA 2. Tabela de multiplicação de $U(20)/H$

Como já sabemos que a operação está adequadamente definida, podemos nos perguntar se o conjunto das classes G/H é um grupo abeliano. A resposta é sim, como veremos agora. Em primeiro lugar, se $a, b, c \in G$ então, por (5),

$$\widehat{a} \star (\widehat{b} \star \widehat{c}) = \widehat{a} \star \widehat{(b \star c)}$$

que é igual à classe de $a \star (b \star c)$ em G/H. Como

$$a \star (b \star c) = (a \star b) \star c,$$

em G, e

$$(\widehat{a} \star \widehat{b}) \star \widehat{c} = \widehat{(a \star b)} \star \widehat{c},$$

que é igual à classe de $(a \star b) \star c$, temos que a operação $\star$ de G/H é associativa. A comutatividade pode ser provada de maneira análoga. O elemento neutro de G/H é $\widehat{e}$, já que

$$\widehat{a} \star \widehat{e} = \widehat{a \star e} = \widehat{a}.$$

Finalmente, se a tem inverso a^{-1} em G, então

$$\widehat{a} \star \widehat{a^{-1}} = \widehat{a \star a^{-1}} = \widehat{e},$$

de modo que $\widehat{a^{-1}}$ é o inverso de $\widehat{a}$ em G/H. Isto não só confirma que G/H é um grupo, como mostra que suas propriedades refletem, de perto, as propriedades de G. Diremos que G/H é o *grupo quociente* de G por H.

Resta-nos, apenas, utilizar a maquinaria desenvolvida nesta seção para provar o teorema de Lagrange.

DEMONSTRAÇÃO DO TEOREMA DE LAGRANGE. Seja G um grupo abeliano finito e denotemos por $\star$ sua operação. Seja H um subgrupo de G. Como a congruência módulo H estabelece uma relação de equivalência em G, temos que

- G é união das classes de equivalência distintas relativamente à congruência módulo H, e
- duas classes de equivalência diferentes têm que ser disjuntas.

Podemos resumir isto escrevendo,

$$G = C_1 \cup \cdots \cup C_t \text{ onde } C_i \cap C_j = \emptyset \text{ se } i \neq j.$$

Mas isto implica que

$$\#G = \#C_1 + \cdots + \#C_t.$$

Portanto, se

$$\#C_1 = \cdots = \#C_t = \#H, \tag{6}$$

então $\#G = t(\#H)$ e o teorema está provado. Assim, basta mostrar que (6) vale.

Em primeiro lugar, $\widehat{e} = H$, de modo que H coincide com uma das classes de equivalência em G/H. Se $a \notin H$, então

$$\widehat{a} = \{a \star x : x \in H\},$$

por (4). Em particular, $\widehat{a}$ não pode ter mais elementos que H. Por outro lado, para que $\#\widehat{a} < \#H$, deveríamos ter

$$a \star x_1 = a \star x_2 \text{ para dois elementso distintos } \; x_1, x_2 \in H.$$

Entretanto, multiplicando $a \star x_1 = a \star x_2$ pelo inverso de a em G, obtemos $x_1 = x_2$. Logo, $\#\widehat{a} = \#H$. Como toda classe em G/H é da forma $\widehat{a}$ para algum $a \in G$, provamos a igualdade (6), o que completa a demonstração do teorema de Lagrange.

5. Exercícios

1. Seja S um subgrupo do grupo aditivo $\mathbb{Z}$, e seja n o menor inteiro positivo contido em S.
   (a) Mostre que todo elemento de S é múltiplo de n.
   (b) Conclua que $S = H_n = \{kn : k \in \mathbb{Z}\}$.

2. Seja G um grupo e $a \in G$ um elemento de ordem k. Mostre que
$$e, a, \ldots, a^{k-1}$$
   são elementos distintos de G.

3. Seja G um grupo abeliano, provido de uma operação $\star$, e sejam $a_1, \ldots, a_n$ elementos de G de ordem $k_1, \ldots, k_n$, respectivamente.
   (a) Mostre que se $r_1 < k_1, \ldots, r_n < k_n$ são inteiros não negativos, então o inverso de $a_1^{r_1} \star \cdots \star a_n^{r_n}$ é $a_1^{k_1 - r_1} \star \cdots \star a_n^{k_n - r_n}$.
   (b) Prove, em detalhes, que $\langle a_1, \ldots, a_n \rangle$ é um subgrupo de G.

4. Seja $n \geq 3$ um inteiro positivo. Dê exemplo de um elemento de $U(n)$, diferente de $\overline{1}$, que não é um gerador de $U(n)$.

5. Seja G um grupo cíclico de ordem n e gerador g.
   (a) Mostre que G é um grupo abeliano.
   (b) Mostre que se d é um inteiro positivo que divide n então G tem um subgrupo (cíclico) de ordem d.
   (c) Mostre que todo subgrupo de G é cíclico.
   (d) Mostre que se k é um inteiro positivo primo com n então g^k também é um gerador de G.
   (e) Use (d) para determinar quantos geradores diferentes G tem.

6. Seja G um grupo. Mostre que se todos os elementos de G, diferentes de e, geram G, então G tem ordem prima.

7. Seja G um grupo abeliano com operação $\star$ e sejam a e b elementos de ordem 2 distintos em G.
   (a) Mostre que $a \star b$ tem ordem 2.
   (b) Mostre que $\{e, a, b, a \star b\}$ é um subgrupo não cíclico de ordem 4 de G.

(c) Mostre que, se G tem m elementos de ordem 2 então o número de subgrupos não cíclicos de ordem 4, distintos, em G é igual a $\binom{m}{2}/3$.

8. Determine todos os subgrupos não cíclicos de ordem 4 de $U(40)$.

9. Seja G um grupo abeliano e S_1 e S_2 subgrupos de G.
   (a) Mostre que $S_1 \cap S_2$ é um subgrupo de G.
   (b) Dê um exemplo em que $S_1 \cup S_2$ não é um subgrupo de G.
   (c) Mostre que se a ordem de S_1 é prima com a ordem de S_2 então $S_1 \cap S_2 = \{e\}$.
   (d) Mostre que o conjunto
   $$S_1S_2 = \{x_1x_2 : x_1 \in S_1 \text{ e } x_2 \in S_2\},$$
   é um subgrupo de G.

10. Seja G um grupo abeliano e suponha que $g_1 \in G$ tem ordem p_1 e que $g_2 \in G$ tem ordem p_2. Mostre que se p_1 e p_2 são primos distintos então g_1g_2 tem ordem p_1p_2.

11. Seja G um grupo abeliano e S_1 e S_2 subgrupos de G. Suponha que S_1 tem ordem p_1 e S_2 tem ordem p_2. Mostre que se p_1 e p_2 são primos distintos, então S_1S_2 é um grupo cíclico de ordem p_1p_2.

12. Sejam G um grupo abeliano e H um subgrupo de G. Prove que a operação em G/H é comutativa.

13. Seja G o grupo multiplicativo formado pelos números racionais da forma $2^k3^m5^n$, onde k, m e n são números inteiros (positivos, negativos ou nulos), e seja S o subgrupo de G gerado por 2^k. Descreva o grupo G/S.

14. Seja G um grupo abeliano e considere o conjunto
   $$S = \{x^2 : x \in G\}.$$
   (a) Mostre que S é um subgrupo de G.
   (b) Mostre que todos os elementos de G/S têm ordem 2.

15. Seja G um grupo abeliano de ordem n e p um primo positivo que divide n. O objetivo desta questão é provar o *Teorema de Cauchy*, segundo o qual G tem que conter um elemento de ordem p. Escreva uma demonstração detalhada deste teorema seguindo o roteiro abaixo:
   (a) Escolha $a_1, \ldots, a_p \in G$. Quantas permutações diferentes a seqüência $(a_1, \ldots, a_p)$ admite?
   (b) Mostre que, como p é primo, o número de permutações da seqüência $(a_1, \ldots, a_p)$ tem que ser um múltiplo de p, *a não ser que*
   $$a_1 = a_2 = \cdots = a_p.$$
   (c) Suponha que $a_1a_2 \cdots a_p = e$, onde e é o elemento neutro de G. Mostre que a_p fica completamente determinado por $a_1, \ldots, a_{p-1}$.

(d) Seja
$$S = \{(a_1, \ldots, a_p) : a_1, \ldots a_p \in G \text{ e } \ a_1 a_2 \cdots a_p = e\}.$$
Use (c) para mostrar que $\sharp S = n^{p-1}$.

(e) Seja
$$S_i = \{(a_1, \ldots, a_p) \in S : a_1 = a_2 = \cdots = a_p\}$$
e $S_d = S \setminus S_i$. Então $S = S_d \cup S_i$, onde a união é disjunta. Mostre que S_i não pode ser vazio exibindo explicitamente um elemento que pertence a S_i.

(f) Mostre, usando (b), que o número de elementos de S_d é um múltiplo de p.

(g) Mostre, usando (f), que o número de elementos de S_i também tem que ser um múltiplo de p. Conclua disto que G tem elementos de ordem p.

Este exercício está baseado em **[McKay 1959]**.

16. Seja G um grupo abeliano e S o subgrupo definido no exercício 14. Use o exercício anterior para provar que a ordem de G/S é uma potência de 2.

Capítulo 3

Anéis, ideais e polinômios

Neste capítulo estudamos uma outra estrutura algébrica: os anéis. Como no caso de grupos, trata-se de uma noção cuja função é apenas a de abarcar vários objetos que surgem em diferentes áreas da matemática. Para nossos propósitos, dois tipos de anéis desempenham um papel especial, os corpos e os anéis de polinômios. Neste capítulo também estendemos, à categoria dos anéis, os conceitos de subgrupo e de grupo quociente.

1. Anéis

A noção de anel surgiu a partir da sistematização das propriedades dos números (inteiros, racionais e reais) realizada pelos matemáticos do século XIX. Entretanto, foi só no início do século XX que estas propriedades foram organizadas na forma das definições gerais que utilizamos até hoje.

Antes de mais nada, usaremos a palavra operação no mesmo sentido em que foi empregada no capítulo 2. Levando isto em conta, dizemos que um conjunto não vazio A é um *anel* se nele estão definidas duas operações, que chamaremos de adição (denotada por $+$) e multiplicação (denotada pela justaposição dos elementos que estão sendo multiplicados), que satisfazem as seguintes propriedades:

- quaisquer que sejam $a, b, c \in A$,

$$\begin{array}{lcl} a+(b+c)=(a+b)+c & \text{e} & a(bc)=(ab)c \\ a+b=b+a & \text{e} & ab=ba \end{array}$$

$$a(b+c)=ab+ac$$

- existem dois elementos especiais em A, denotados por 0 e 1, tais que, para todo $a \in A$,

$$a+0=a \qquad \text{e} \qquad a1=a$$

- para todo $a \in A$ existe um elemento $-a$ tal que

$$a+(-a)=0.$$

Listamos algumas das propriedades da adição e da multiplicação lado a lado para chamar atenção sobre a sua semelhança. Observe que estamos exigindo que a operação de multiplicação de um anel seja comutativa; isto é, que $ab = ba$. Na maior parte dos livros de álgebra que você consultar o objeto que acabamos de definir será chamado um *anel comutativo*. Entretanto, não há razão para adotar esta nomenclatura aqui já que todos os anéis que consideramos neste livro são comutativos.

Os elementos especiais 0 e 1 são conhecidos como *elementos neutros* da adição e multiplicação, respectivamente. O elemento $-a$, definido na última propriedade, é chamado de *simétrico* de a. Esta terminologia é reminiscente do caso em que $A = \mathbb{Z}$; neste caso, dispondo os inteiros ao longo de uma reta da maneira usual, temos que a e $-a$ são simétricos em relação à origem. Observe que o conjunto A é um grupo abeliano relativamente à soma, mas não à multiplicação.

Os anéis são abundantes entre os objetos algébricos que nos são familiares. Os exemplos mais simples incluem os números inteiros, os racionais, os reais, os complexos e o conjunto $\mathbb{Z}_n$, das classes de inteiros módulo n. Outro exemplo familiar, que estudaremos com muito cuidado, é o conjunto dos polinômios em uma variável, com as operações usuais de adição e multiplicação de polinômios.

Entretanto, nem todo conjunto onde estão definidas uma soma e uma multiplicação é um anel. O exemplo mais natural é o conjunto $\mathbb{N}$ dos inteiros não negativos. Neste caso, todas as propriedades que definem um anel são satisfeitas exceto uma, a existência do simétrico.

Alguns tipos especiais de anéis são tão importantes, e ocorrem com tanta freqüência, que têm um nome especial. Um desses são os corpos. Dizemos que um anel K é um *corpo*, se para cada elemento $0 \neq a \in K$, existe um elemento $b \in K$ tal que $ab = 1$. O elemento b é conhecido como o *inverso* de a e será denotado por a^{-1} ou $1/a$. Em outras palavras, K é um corpo se, e somente se, $K \setminus \{0\}$ é um grupo multiplicativo. Exemplos de corpos incluem $\mathbb{Q}$, $\mathbb{R}$ e $\mathbb{C}$. Entretanto, por causa das aplicações à teoria de números, o exemplo mais importante de corpo a ser usado neste livro é $\mathbb{Z}_p$, quando p é um número primo. Ao contrário de $\mathbb{Q}$, $\mathbb{R}$ e $\mathbb{C}$, o corpo $\mathbb{Z}_p$ tem um número finito de elementos.

Observe que $\mathbb{Z}$ não é um corpo porque os únicos inteiros que têm inverso em $\mathbb{Z}$ são 1 e -1. Entretanto, $\mathbb{Z}$ se enquadra em uma outra classe importante de anéis, os domínios. Um anel A é um *domínio* se sempre que $ab = 0$, para dois elementos $a, b \in A$, então $a = 0$ ou $b = 0$. Um exemplo de anel que *não* satisfaz esta propriedade é $\mathbb{Z}_6$:

$$\overline{2} \text{ e } \overline{3} \text{ são classes não nulas em } \mathbb{Z}_6, \text{ mas } \overline{2} \cdot \overline{3} = \overline{0}.$$

Na verdade, $\mathbb{Z}_n$ só pode ser um domínio se n for primo. Só que, neste caso, $\mathbb{Z}_n$ já é um corpo, como mostra o exercício 1.

Na próxima seção consideramos em detalhes a construção dos anéis de polinômios e suas propriedades básicas.

2. Anéis de polinômios

Nesta seção estaremos supondo que A é um anel qualquer. Construiremos a seguir o anel de polinômios em uma variável com coeficientes em A. Entretanto, se você preferir, pode imaginar, em uma primeira leitura, que A é $\mathbb{Q}$, ou outro anel que lhe seja familiar.

Seja x um símbolo, tradicionalmente conhecido como *variável* ou *indeterminada*. Um *polinômio f na variável x, com coeficientes em A*, é uma expressão da forma

$$f = a_n x^n + \cdots + a_2 x^2 + a_1 x + a_0, \tag{7}$$

onde n é um inteiro não negativo e $a_0, \ldots, a_n$ são elementos de A. Dizemos que a_j é o *coeficiente* de f em grau j. Se $a_n \neq 0$ e $a_j = 0$ para todo $j > n$, então o inteiro n é o *grau* de f e a_n é o seu *coeficiente líder*. Eles serão denotados por $\mathrm{grau}(f)$ e $\mathrm{ld}(f)$, respectivamente. Um polinômio é *mônico* se o seu coeficiente líder for igual a 1.

Denotaremos o polinômio em (7), indistintamente, como f ou $f(x)$, dependendo se precisamos ou não chamar a atenção para a variável que está sendo utilizada. O conjunto de todos os polinômios na variável x, com coeficientes no anel A, será denotado por $A[x]$.

A esta altura, um polinômio não passa de símbolos encadeados, formando uma espécie de 'palavra'. Para dar vida a estes objetos precisamos explicar como interagem entre si. O primeiro passo é definir a igualdade de polinômios. Seja

$$h = b_m x^m + \cdots + b_2 x^2 + b_1 x + b_0,$$

um outro polinômio. Dizemos que $f = h$ se $m = n$ e se $a_i = b_i$ para cada $0 \leq i \leq n$. Uma conseqüência imediata desta definição é que, se $m < n$, então os polinômios h e

$$0x^n + \cdots + 0x^{m+2} + 0x^{m+1} + b_m x^m + \cdots + b_1 x + b_0,$$

são iguais. Isto significa que sempre podemos supor que dois polinômios tenham o mesmo número de coeficientes, *desde que não estejamos fazendo nenhuma hipótese quanto a estes coeficientes serem ou não nulos!* Usaremos isto, de agora em diante, sem maiores cerimônias.

Seja

$$g = b_n x^n + \cdots + b_2 x^2 + b_1 x + b_0,$$

um polinômio em $A[x]$. Definimos a soma $f + g$ como sendo o polinômio

$$f + g = (a_n + b_n)x^n + \cdots + (a_1 + b_1)x + (a_0 + b_0). \tag{8}$$

De posse desta definição, podemos dizer que f é igual à soma dos polinômios $a_0, a_1 x, \ldots, a_n x^n$. Não esqueça que nossa convenção de permitir coeficientes nulos faz com que a fórmula da soma $f + g$ seja completamente geral. Por outro lado, se você assumir que $a_n \neq 0$ ou $b_n \neq 0$ em (8), então o grau de $f + g$ é, no máximo, n. Em outras palavras, temos a seguinte fórmula

$$\mathrm{grau}(f + g) \leq \max\{\mathrm{grau}(f), \mathrm{grau}(g)\}.$$

Para descobrir como o produto fg deve ser definido, vamos supor que a multiplicação de polinômios deve ser distributiva, e que

$$(a_i x^i)(b_j x^j) = a_i b_j x^{i+j}, \tag{9}$$

onde o produto $a_i b_j$ é calculado no anel A. Levando isto em conta temos que

$$fg = \sum_{i,j} a_i b_j x^{i+j}. \tag{10}$$

Mas esta fórmula não é muito satisfatória porque a expressão resultante não está na forma da equação (7). Por exemplo, aplicando a fórmula ao produto $(2x^2 + 3x)(x^3 + 3x^2 + 1)$, obtemos

$$2x^5 + 6x^4 + 2x^2 + 3x^4 + 9x^3 + 3x.$$

Entretanto, para converter (10) para a forma padrão basta agrupar os termos de mesmo grau. Fazendo isto, no exemplo, obtemos

$$(2x^2 + 3x)(x^3 + 3x^2 + 1) = 2x^5 + 9x^4 + 9x^3 + 2x^2 + 3x.$$

Um cálculo simples mostra que as expressões para os coeficientes dos três termos de menor grau de fg são da forma

coeficiente de x^0: $a_0 b_0$;
coeficiente de x^1: $a_0 b_1 + a_1 b_0$;
coeficiente de x^2: $a_0 b_2 + a_1 b_1 + a_2 b_0$.

Isto sugere que a fórmula geral para o coeficiente de x^k deve ser

$$\sum_{i=0}^{n} a_i b_{k-i} \tag{11}$$

como se pode provar por indução em k.

Ao escrever a fórmula (11), estamos subentendendo que alguns dos coeficientes podem ser nulos, mesmo se estão mais à frente do que o último coeficiente não nulo de f ou g. Portanto, se f tem grau n e g tem grau m, então (11) implica que:

- os coeficientes de fg de grau maior que $n + m$ são todos nulos;
- o coeficiente de fg de grau $n + m$ é $a_n b_m$.

Como estamos supondo que f tem grau n e g tem grau m, os coeficientes a_n e b_m não podem ser nulos. Se, além disto, A for um domínio, então $a_n b_m \neq 0$. Em particular, concluímos que, se A for um domínio,

$$\text{grau}(fg) = \text{grau}(f) + \text{grau}(g).$$

Antes de passar à divisão de polinômios, precisamos determinar qual o custo incorrido em efetuar as operações de soma e multiplicação de polinômios. De acordo com (8), para calcular $f + g$, apenas somamos os coeficientes dos termos de mesmo grau, num total de n adições de elementos do anel de base A. Para obter um resultado mais preciso, precisamos conhecer o custo de calcular somas em A. Como isto varia de anel para anel, suporemos que $A = \mathbb{Z}_m$, porque este é o caso que usaremos no capítulo 5. Neste caso, o custo da soma é da

ordem de $O(\log_2 m)$, como vimos na seção 2 do capítulo 1. Como precisamos efetuar n somas de coeficientes, o custo total necessário para determinar $f+g$ será $O(n(\log_2 m))$.

No caso da multiplicação, usamos a fórmula (10). Observe que as potências da variável x apenas marcam a posição dos coeficientes no vetor que descreve o polinômio. Portanto, calcular com potências de x não requer nenhuma operação aritmética.

Como para a adição, suporemos que o anel de coeficientes é $\mathbb{Z}_m$. Para determinar o produto de dois termos como em (9) precisamos calcular o produto dos coeficientes, ao custo $O((\log_2 m)^2)$. Como não há mais que n^2 produtos de termos a calcular, o custo total em efetuá-los será de $O(n^2(\log_2 m)^2)$. Contudo, tendo multiplicado os termos dois a dois, ainda precisamos somar os coeficientes de dois produtos cujo grau coincide. Segundo (11) há, no máximo, k somas a calcular para cada $0 \leq k \leq 2n$. Portanto, o número de somas não pode ser maior do que

$$0 + 1 + \cdots + 2n = \frac{2n(2n+1)}{2}.$$

Levando isto em conta, concluímos que o custo de efetuar estas somas é da ordem de $O(n^2(\log_2 m))$. Portanto, o custo total necessário para calcular fg é $O(n^2(\log_2 m)^2)$. Convém reunir estes resultados em uma proposição para facilitar a referência futura.

PROPOSIÇÃO 3.1. *Sejam f e g polinômios em $\mathbb{Z}_m[x]$ e seja d o máximo entre os graus de f e g. Então:*

- *$f+g$ pode ser calculado ao custo $O(d(\log_2 m))$;*
- *fg pode ser calculado ao custo $O(d^2(\log_2 m)^2)$.*

3. Divisão de polinômios

Já discutimos adição e multiplicação de polinômios, falta-nos falar sobre a divisão de polinômios. Para simplificar a discussão assumiremos, de agora em diante, que estamos trabalhando com polinômios com coeficientes em um domínio A.

Como no caso dos números inteiros, trataremos da divisão com resto. Em outras palavras, ao dividir um polinômio f por um polinômio g, esperamos encontrar dois novos polinômios, o quociente q e o resto r, de modo que $f = gq + r$. Além disto, queremos que o resto seja, de alguma forma, "menor" que o divisor. Faremos isto exigindo que r seja nulo, ou que tenha grau menor que o grau de g. Uma maneira de provar a existência de q e r é construir um algoritmo para calculá-los. Nosso algoritmo será recursivo, e reduzirá o grau do dividendo f até que seja menor que o grau de g.

Digamos que

$$f = a_n x^n + \cdots + a_1 x + a_0 \text{ e } g = b_m x^m + \cdots + b_1 x + b_0$$

são polinômios com coeficientes em um domínio A, para os quais $a_n \neq 0$ e $b_m \neq 0$. Se, por acaso, $f = 0$, então o resto e o quociente também serão nulos, já que 0 é divisível por qualquer polinômio. Por outro lado, se $n < m$, então o resto será o próprio f, e o quociente será zero. Podemos, assim, supor que $f \neq 0$ e que $n \geq m$. Neste caso, queremos subtrair de f um múltiplo de g de modo a anular o termo de maior grau de f. Para isto precisamos fazer duas coisas:

(1) compensar a diferença entre o grau de g e o de f multiplicando g por x^{n-m};

(2) compensar os coeficientes, multiplicando g por a_n/b_m.

É claro que (1) não oferece nenhum dificuldade, mas para efetuar (2) precisamos poder dividir a_n por b_m em A. Entretanto, isto nem sempre é possível em um domínio. Por isso, precisamos fazer, neste ponto, a hipótese adicional de que b_m é invertível em A. Observe que se A for um corpo isto significa apenas que $b_m \neq 0$. Como de costume, denotaremos o inverso de b_m em A por $1/b_m$.

Supondo, então, que b_m tem inverso em A, efetuamos as operações (1) e (2), calculando $f - (a_n/b_m)x^{n-m}g$, cujo coeficiente de grau n é 0. Portanto, $f - (a_n/b_m)x^{n-m}g$ tem grau menor que o grau de f. O algoritmo consiste na repetição deste passo até que sobre um polinômio de grau menor que g. Mais precisamente, temos o seguinte conjunto de instruções.

Algoritmo de divisão para polinômios em uma variável

Entrada: polinômios $f, g \in A[x]$, onde A é um domínio, e o coeficiente de maior grau de g é invertível em A.

Saída: polinômios $q, r \in A[x]$ tais que

$$f = gq + r \text{ e } r = 0 \text{ ou } r \text{ tem grau menor que o grau de } g$$

Primeira etapa: Inicializa $F = f$, $Q = 0$;

Segunda etapa: Se $F = 0$ ou F tem grau menor que o grau de g, então páre e imprima: "o resto é F e o quociente é Q".

Terceira etapa: Faça

$$F = F - \frac{\mathrm{ld}(F)}{\mathrm{ld}(g)}x^d g \text{ e } Q = Q + \frac{\mathrm{ld}(F)}{\mathrm{ld}(g)}x^d,$$

onde $d = \mathrm{grau}(F) - \mathrm{grau}(g)$, e volte à segunda etapa.

Como no caso de inteiros, apresentamos a divisão de polinômios em uma tabela, como ilustra o exemplo da próxima página. Neste exemplo, o quociente é $6x^2 + 7x + 8$ e o resto é $-6x^3 - 13x^2 - 15x - 7$.

Se o resto da divisão de $f(x)$ por $g(x)$ em $A[x]$ for zero, diremos que $g(x)$ é um *divisor* ou *fator* de $f(x)$. Em outras palavras, $g(x)$ é um divisor de $f(x)$ se existe um polinômio $q(x) \in A[x]$ tal que $f(x) = g(x)q(x)$. As propriedades da divisibilidade de polinômios são muito semelhantes às da divisibilidade de inteiros, como ilustra o exercício 2.

$$
\begin{array}{rl|l}
6x^6 + 7x^5 + 8x^4 + 1 & & x^4 + x + 1 \\ \cline{3-3}
7x^5 + 8x^4 - 6x^3 - 6x^2 + 1 & & 6x^2 + 7x + 8 \\
8x^4 - 6x^3 - 13x^2 - 7x + 1 & & \\
-6x^3 - 13x^2 - 15x - 7 & &
\end{array}
$$

TABELA 1. Divisão de $6x^6 + 7x^5 + 8x^4 + 1$ por $x^4 + x + 1$

Precisamos provar que o algoritmo funciona; quer dizer, mostrar que sempre pára e que calcula corretamente o que foi especificado na entrada. Para isto, denote por F_i e Q_i os valores das variáveis F e Q no i-ésimo laço da aplicação do algoritmo. Se escrevermos $F_0 = f$ para o valor de F no início da computação, teremos que

$$\text{grau}(F_0) > \text{grau}(F_1) > \text{grau}(F_2) > \cdots \geq 0,$$

pois

$$F_{i+1} = F_i - \frac{\text{ld}(F_i)}{\text{ld}(g)} x^{(\text{grau}(F_i) - \text{grau}(g))} g$$

foi construído de modo que houvesse cancelamento do termo líder de F_i. Portanto, é evidente que o algoritmo tem que parar. Por outro lado, se o algoritmo pára no k-ésimo passo, então

$$r = 0 \text{ ou } \text{grau}(F_k) < \text{grau}(g),$$

Mas temos, por indução em i, que $F_i = f - Q_i g$ para todo $0 \leq i \leq k$. Logo, tomando $r = F_k$ e $q = Q_k$ obtemos $f = qg + r$ e $r = 0$ ou r tem grau menor que o grau de g, como havia sido especificado na saída. É conveniente enunciar o resultado do algoritmo sob a forma de um teorema.

TEOREMA 3.2. *Sejam f e g polinômios em $A[x]$. Se o coeficiente líder de g é invertível, então existem polinômios $q, r \in A[x]$, tais que*

$$f = qg + r \text{ e } r = 0 \text{ ou } \text{grau}(r) < \text{grau}(g).$$

Além disto, os polinômios q e r ficam unicamente determinados por estas condições.

A existência de q e r foi provada anteriormente, uma vez que já conhecemos um algoritmo que nos permite calculá-los. Resta-nos apenas provar a unicidade. Para isto, suponhamos que duas pessoas recebem polinômios f e g em $A[x]$. Provar a unicidade do quociente e do resto significa que, independente de como estas pessoas calculem q e r, se as condições do teorema são satisfeitas então ambas obterão o mesmo quociente e o mesmo resto. Suponhamos, então, que a pessoa 1 calcula um quociente q_1 e um resto r_1 tais que

$$f = q_1 g + r_1 \text{ e } r_1 = 0 \text{ ou } \text{grau}(r_1) < \text{grau}(g),$$

e que a pessoa 2 calcula um quociente q_2 e um resto r_2 com

$$f = q_2 g + r_2 \text{ e } r_2 = 0 \text{ ou } \operatorname{grau}(r_2) < \operatorname{grau}(g).$$

Subtraindo as expressões para f obtidas por 1 e 2, verificamos que

$$(q_2 - q_1)g = r_1 - r_2. \tag{12}$$

Se $r_1 - r_2 = 0$, então $q_2 - q_1 = 0$, e ambas as pessoas obtiveram o mesmo quociente e o mesmo resto, como queríamos provar. Portanto, basta mostrar que $r_1 - r_2 \neq 0$ não pode acontecer. Mas $r_1 - r_2 \neq 0$ implica que $r_1 \neq 0$ ou $r_2 \neq 0$. Em qualquer dos dois casos, $\operatorname{grau}(g) > \operatorname{grau}(r_1 - r_2)$. Contudo, a equação (12) implica que $r_1 - r_2 \neq 0$ é múltiplo de g, de modo que não pode ter grau menor que $\operatorname{grau}(g)$. Esta contradição mostra que $r_1 - r_2 \neq 0$ não pode acontecer, como queríamos mostrar.

Podemos usar o algoritmo de divisão para relacionar as raízes de um polinômio com os seus fatores de grau um. Digamos que A seja um domínio e que

$$f(x) = a_n x^n + \cdots + a_1 x + a_0 \in A[x].$$

Um elemento $\alpha \in A$ é uma *raiz* de $f(x)$ se

$$f(\alpha) = a_n \alpha^n + \cdots + a_1 \alpha + a_0 = 0.$$

LEMA 3.3. *Seja A um domínio e $f \in A[x]$. Então $\alpha \in A$ é uma raiz de f se, e somente se, $x - \alpha$ divide f.*

DEMONSTRAÇÃO. Suponhamos que $\alpha \in A$ é raiz de f. Dividindo f por $x - \alpha$ podemos escrever

$$f(x) = (x - \alpha)q(x) + r. \tag{13}$$

Mas $r = 0$, ou r tem grau menor que 1, que é o grau do divisor $x - \alpha$. Em qualquer dos dois casos, r é uma constante. Substituindo x por α em (13), e levando em conta que α é raiz de $f(x)$, temos

$$0 = f(\alpha) = (\alpha - \alpha)q(\alpha) + r = r,$$

já que r é uma constante e não depende de x. Logo, $f(x) = (x - \alpha)q(x)$; isto é, $x - \alpha$ divide de f. A recíproca é imediata.

Também podemos usar o algoritmo de divisão para calcular o máximo divisor comum entre dois polinômios. De fato, é relativamente fácil adaptar tanto o procedimento do algoritmo euclidiano estendido, quanto a sua demonstração, para o anel de polinômios sobre um corpo. Só não fazemos isto aqui porque não teremos nenhuma aplicação para o algoritmo neste livro. Para mais detalhes consulte **[von zur Gathen e Gerhard 1999, seção 3.1]** ou **[Cohen 2003, seção 4.2]**.

4. Ideais

Nada nos impede de usar o fato de que um anel A é um grupo abeliano com respeito à operação de soma para construir o grupo quociente de A/S com respeito a um subgrupo aditivo S de A. O problema é que o fato de A estar dotado de uma multiplicação aumenta as nossas expectativas. Já não estaremos contentes em fazer de A/S um mero subgrupo aditivo; queremos que A/S herde de algum modo a multiplicação de A e seja um anel. Afinal de contas, foi exatamente isto que fizemos quando construímos $\mathbb{Z}_n$, e funcionou.

Vejamos o que acontece se tentamos definir um produto em A/S. Sejam $\widehat{a}$ e $\widehat{b}$ classes em A/S. Tanto a definição da operação de um grupo quociente, quanto a de multiplicação em $\mathbb{Z}_n$ sugerem que devemos pôr

$$\widehat{a} \cdot \widehat{b} = \widehat{ab}, \tag{14}$$

onde ab indica o produto de a e b como elementos de A. O problema é que esta multiplicação de classes pode não estar bem definida. Para ver porque, suponha que multiplicamos $\widehat{a}$ por $\widehat{0}$. Segundo a definição (14), devemos obter

$$\widehat{a} \cdot \widehat{0} = \widehat{a0} = \widehat{0}.$$

Contudo, $\widehat{0} = \widehat{s}$, qualquer que seja $s \in S$. Se a fórmula (14) representar uma operação bem definida, então usando s, ao invés de 0, como representante da classe, devemos obter o mesmo resultado final. Mas,

$$\widehat{a} \cdot \widehat{s} = \widehat{as},$$

e, para esta classe ser sempre igual a $\widehat{0}$, precisamos que

$$as \in S \text{ para todo } s \in S \text{ e todo } a \in A. \tag{15}$$

Só que isto não é verdade para todo subgrupo. Por exemplo, se S for o subgrupo cíclico gerado por 1 em $\mathbb{Q}[x]$, então $S = \mathbb{Z}$ e a propriedade (15) é dramaticamente falsa neste caso. Afinal, $f(x) \cdot 1 \notin S$ sempre que $f(x)$ for um polinômio de grau maior que zero.

Em vista disso, há três saídas. A primeira é decidir que estamos querendo demais, e desistir. A segunda é tentar definir a multiplicação de outra maneira. A terceira, é declarar que para ter, além da soma, uma multiplicação bem definida em A/S, precisamos que S seja um subgrupo aditivo que satisfaz uma propriedade extra; a saber (15). Nada garante *a priori* que esta última saída vá funcionar. Bastaria, por exemplo, que muito poucos subgrupos satisfizessem (15), para que a utilidade desta alternativa ficasse extremamente comprometida. Para sorte nossa, isto não acontece, e é exatamente esta terceira via a que vamos adotar.

Formalizando o que fizemos acima, diremos que um subconjunto não vazio I de um anel A é um *ideal* de A se

(i) $x + y \in I$ para todo $x, y \in I$, e
(ii) $ax \in I$, quaisquer que sejam $a \in A$ e $x \in I$.

Talvez sua primeira reação, ao olhar estas propriedades, seja:

> "Mas um ideal não tinha que ser um subgrupo aditivo? Como, então, podem estas duas propriedades não dizer nada sobre o 0 ou sobre os simétricos?"

Acontece que, na definição de ideal, temos a propriedade (ii), que vem em nosso socorro. Afinal, se $x \in I$ então, por (ii), $(-1)x = -x \in I$. Segue disto e de (i) que $x + (-x) = 0 \in I$. Portanto, (i) e (ii) implicam que I é um subgrupo aditivo de A.

Como todo ideal é um subgrupo aditivo do anel A, não é muito surpreendente que o menor ideal de A seja $\{0\}$, e que o maior seja A. Se $A = K$ for um corpo, então estes são os únicos ideais de A. Par ver porque, suponha que I seja um ideal não nulo de K. Então, I contém um elemento $a \neq 0$. Como K é um corpo, $a^{-1} \in K$. Da condição (ii) segue então que,

$$x = x(aa^{-1}) \in I,$$

para todo $x \in K$. Logo $I = K$. Mostramos, portanto, que todo ideal não nulo de K é igual ao próprio K. A recíproca deste resultado também é verdadeira: se os únicos ideais de um anel A são $\{0\}$ e A então A é um corpo. A demonstração deste fato fica por sua conta; veja exercício 4.

Se A não for um corpo, podemos construir ideais mais interessantes a partir de geradores. Digamos que queremos construir o menor ideal I que contém $a_1, \ldots, a_n \in A$. Por (ii) temos que

$$a_1x_1, \ldots, a_nx_n \in I, \text{ quaisquer que sejam } x_1, \ldots, x_n \in A.$$

Mas, por (i), a soma destes elementos também pertence a I. Logo,

$$a_1x_1 + \cdots + a_nx_n \in I \text{ quaisquer que sejam } x_1, \ldots, x_n \in A.$$

Acontece que uma verificação simples mostra que o conjunto

$$\{a_1x_1 + \cdots + a_nx_n : x_1, \ldots, x_n \in A\},$$

é um ideal de A e, portanto, tem que ser igual a I. Definimos, assim, o *ideal gerado* por $a_1, \ldots, a_n$, que denotaremos por $(a_1, \ldots, a_n)$ para diferenciá-lo do subgrupo $\langle a_1, \ldots, a_n \rangle$ gerado pelos mesmos elementos. Um caso importante é o do ideal que é gerado por um único elemento. Estes ideais, chamados de *principais*, desempenham na teoria de anéis um papel semelhante ao dos grupos cíclicos na teoria de grupos.

Para deixar bem clara a diferença entre ideais e subgrupos aditivos de um anel, vamos dar dois exemplos no anel $\mathbb{Z}_n[x]$. Em primeiro lugar, multiplicar qualquer elemento $f(x) \in \mathbb{Z}_n[x]$ por n tem o efeito de reduzi-lo a $\overline{0}$. Portanto, o subgrupo aditivo de $\mathbb{Z}_n[x]$ gerado pelo polinômio x é

$$\langle x \rangle = \{\overline{0}, x, \overline{2}x, \ldots, \overline{n-1}x\}.$$

Contudo, o ideal gerado por x contém todos os "múltiplos polinomiais" de x; quer dizer, todos os polinômios da forma $xf(x)$ onde $f(x)$ é um polinômio qualquer de $\mathbb{Z}_n[x]$. Mas estes são, exatamente, os polinômios cujo termo constante é zero. Isto é,

$$(x) = \{f(x) \in \mathbb{Z}_n[x] : f(0) = 0\}.$$

Como $x, x^2, x^3, \cdots \in (x)$, então (x) é um conjunto infinito; ao passo que $\langle x \rangle$ tem apenas n elementos.

A estrutura de um ideal em um anel pode ser, em geral, muito complicada. Para sorte nossa, isto não acontece no caso do anel que mais nos interessa, o anel de polinômios sobre um corpo, como mostra o próximo teorema.

TEOREMA 3.4. *Todo ideal do anel de polinômios em uma variável sobre um corpo é principal.*

É importante frisar que este resultado é falso se os coeficientes dos polinômios não estiverem sobre um corpo. Veja o exercício 6 para mais detalhes. Por outro lado, este teorema é o análogo para anéis de polinômios do fato de que todo subgrupo de $\mathbb{Z}$ é cíclico; cf. exercício 1 do capítulo 2.

DEMONSTRAÇÃO. Seja K um corpo e seja I um ideal de $K[x]$. Tanto $\{0\}$ como $K[x]$ são gerados por um elemento, de modo que podemos supor que $(0) \subset I \subset K[x]$. A estratégia que adotaremos para mostrar que I tem um gerador pode ser dividida em duas partes:

(i) escolher um candidato g a gerador de I;
(ii) mostrar que o resto da divisão de cada elemento de I por g é zero.

Note que (ii) implica que $I \subseteq (g)$. Como $g \in I$, por hipótese, então temos a igualdade $I = (g)$, e o teorema estará provado.

O primeiro desafio consiste em escolher g de maneira adequada. Mas queremos que g divida cada elemento de I, e isto só pode acontecer se tiver o menor grau possível entre os elementos de I. Assim, escolheremos g como sendo um polinômio *não nulo* qualquer de I cujo grau é o menor possível. Note que, em princípio, podem existir vários polinômios distintos satisfazendo esta propriedade.

Para provar que g divide cada elemento de I, tome $f \in I$ e divida-o por g, obtendo

$$f = gq + r \text{ onde } \; r = 0 \text{ ou } \; \text{grau}(r) < \text{grau}(g).$$

Como $f, g \in I$, então $f - qg \in I$, já que I é um ideal de $K[x]$. Assim, se $r \neq 0$ teríamos que

$$r \in I \text{ e } r \text{ tem grau menor que } g,$$

o que contradiz a escolha de g. Portanto, $r = 0$, o que prova (ii) e completa a demonstração do teorema.

5. Anéis quocientes

É hora de voltar aos quocientes, com os quais iniciamos a seção anterior. Digamos que A seja um anel e I seja um ideal de A. Como I é um subgrupo do grupo aditivo de A, podemos construir o grupo quociente A/I, que estará

automaticamente dotado de uma adição. De fato, se $a, b \in A$, então denotando por $\widehat{a}$ e $\widehat{b}$ as classes correspondentes em A/I, teremos que

$$\widehat{a} + \widehat{b} = \widehat{a+b}.$$

Toda a nossa discussão anterior leva a crer que a multiplicação deva ser definida em A/I por

$$\widehat{a} \cdot \widehat{b} = \widehat{ab}.$$

Para verificar que esta fórmula nos dá uma multiplicação bem definida, vamos supor que $a' \in \widehat{a}$ e que $b' \in \widehat{b}$ são outros representantes desta mesma classe. Isto significa que

$$a' = a + s \text{ e } b' = b + t, \text{ onde } s, t \in I.$$

Multiplicando a' por b' obtemos

$$a'b' = (a+s)(b+t) = ab + (bs + at + st).$$

Mas I é um ideal de A, de modo que $bs + at + st \in I$. Portanto, $\widehat{a'b'} = \widehat{ab}$, o que comprova que a multiplicação está bem definida. Deixaremos por sua conta verificar que estas operações de adição e multiplicação fazem de A/I um anel.

Precisaremos usar esta construção, no capítulo 5, no caso em que A é o anel de polinômios sobre um corpo K. Por isso analisaremos este caso em detalhes a seguir. Em primeiro lugar, pelo teorema 3.4, todo ideal de $K[x]$ é principal. Portanto, basta analisar o quociente de $K[x]$ pelo ideal principal gerado por um polinômio $g \in K[x]$. Queremos fazer duas coisas:

- caracterizar as classes de equivalência de $K[x]/(g)$;
- determinar uma fórmula geral para a multiplicação de classes.

Para achar uma forma geral para as classes usaremos o algoritmo de divisão de $K[x]$. Seja $f \in K[x]$. Se f tem grau maior que n, podemos dividi-lo por g, obtendo

$$f = gq + r \text{ onde } r = 0 \text{ ou } r \text{ tem grau menor que } n.$$

Como $f - r = gq \in (g)$, concluímos que $\widehat{f} = \widehat{r}$ em $K[x]$. Mostramos, assim, que toda classe de $K[x]/(g)$ tem um representante cujo grau é menor que n. Além disso, a unicidade do resto da divisão de polinômios mostra que cada classe de equivalência só pode conter um polinômio mônico de grau menor que n. Portanto, basta determinar um polinômio mônico em uma classe de equivalência de $K[x]/(g)$ para caracterizá-la completamente.

Como os elementos de $K[x]$ podem ser escritos como combinação linear das potências de x, então os elementos $K[x]/(g)$ podem ser escritos como combinação das potências de $\widehat{x}$, que é a classe de x módulo (g). Portanto, pela propriedade distributiva, basta saber multiplicar potências de $\widehat{x}$ para saber multiplicar duas classes. Acontece que, por causa da caracterização das classes que vimos anteriormente, queremos nos restringir a representantes de grau menor que n. Assim, toda vez que o produto de duas potências de $\widehat{x}$ tiver grau total maior ou igual a n, teremos que escrevê-lo como combinação de

potências de grau menor que n. Como $x^m = x^{m-1}x$, basta calcular $\widehat{x}^n$ para saber como reduzir todas as potências de $\widehat{x}$ cujo expoente é maior que n. Mas se $g(x) = x^n + a_{n-1}x^{n-1} + \cdots + a_1x + a_0$, então segue de $\widehat{g} = \widehat{0}$ que

$$\widehat{x} \cdot \widehat{x}^{n-1} = \widehat{x}^n = -a_{n-1}\widehat{x}^{n-1} - \cdots - a_1\widehat{x} - a_0. \tag{16}$$

Para podermos dar uma descrição mais precisa das classes e da multiplicação, precisamos restringir ainda mais nosso exemplo. Observe que se K é um corpo infinito, então existem infinitos polinômios mônicos de grau k para cada inteiro $k \geq 0$. A caracterização das classes de $K[x]/(g)$ mostra, então, que este anel quociente tem um número infinito de elementos distintos. Por outro lado, se K é um corpo finito, então o número de polinômios de um dado grau em $K[x]$ será finito. Como cada classe de $K[x]/(g)$ é caracterizada por um polinômio de grau menor que n, podemos concluir que o anel quociente $K[x]/(g)$ é finito neste caso. Em particular, podemos fazer uma lista completa de todas classes de $K[x]/(g)$, quando K é um corpo finito.

Para poder listar sem muita dificuldade as classes de equivalência, escolheremos $K = \mathbb{Z}_2$ e $g(x) = x^2 + x + 1$. Neste caso, o anel quociente $\mathbb{Z}_2[x]/(x^2 + x + 1)$ será igual a

$$\{\widehat{0}, \widehat{1}, \widehat{x}, \widehat{x} + \widehat{1}\}.$$

Usando o que já aprendemos, podemos calcular somas e produtos de classes de $\mathbb{Z}_2[x]/(x^2 + x + 1)$. Por exemplo,

$$(\widehat{x} + \widehat{1})^2 = \widehat{x}^2 + \widehat{1}^2 + \widehat{2}\widehat{x} = \widehat{x}^2 + \widehat{1},$$

pois $2f = 0$, para qualquer polinômio $f \in \mathbb{Z}_2[x]$. Dito de outro modo, $f = -f$ em $\mathbb{Z}_2[x]$. Levando isto em conta, a equação (16) nos dá $\widehat{x}^2 = \widehat{x} + \widehat{1}$, neste caso. Daí,

$$(\widehat{x} + \widehat{1})^2 = \widehat{x}^2 + \widehat{1} = (\widehat{x} + \widehat{1}) + \widehat{1} = \widehat{x}.$$

Como o número de classes, neste exemplo, é pequeno, fica fácil calcular todas as possibilidades para suas somas e produtos. Os resultados destes cálculos estão resumidos nas tabelas 2 e 3. Estas tabelas devem ser interpretadas de maneira análoga às tabelas de grupos do capítulo 2.

$+$	$\widehat{0}$	$\widehat{1}$	$\widehat{x}$	$\widehat{x}+\widehat{1}$
$\widehat{0}$	$\widehat{0}$	$\widehat{1}$	$\widehat{x}$	$\widehat{x}+\widehat{1}$
$\widehat{1}$	$\widehat{1}$	$\widehat{0}$	$\widehat{x}+\widehat{1}$	$\widehat{x}$
$\widehat{x}$	$\widehat{x}$	$\widehat{x}+\widehat{1}$	$\widehat{0}$	$\widehat{1}$
$\widehat{x}+\widehat{1}$	$\widehat{x}+\widehat{1}$	$\widehat{x}$	$\widehat{1}$	$\widehat{0}$

TABELA 2. Adição em $\mathbb{Z}_2[x]/(x^2 + x + 1)$

Uma inspecção rápida da tabela 3 mostra que todos os elementos não nulos do anel $\mathbb{Z}_2[x]/(x^2 + x + 1)$ têm inverso multiplicativo. Em outras palavras,

$\cdot$	$\widehat{0}$	$\widehat{1}$	$\widehat{x}$	$\widehat{x}+\widehat{1}$
$\widehat{0}$	$\widehat{0}$	$\widehat{0}$	$\widehat{0}$	$\widehat{0}$
$\widehat{1}$	$\widehat{0}$	$\widehat{1}$	$\widehat{x}$	$\widehat{x}+\widehat{1}$
$\widehat{x}$	$\widehat{0}$	$\widehat{x}$	$\widehat{x}+\widehat{1}$	$\widehat{1}$
$\widehat{x}+\widehat{1}$	$\widehat{0}$	$\widehat{x}+\widehat{1}$	$\widehat{1}$	$\widehat{x}$

TABELA 3. Multiplicação em $\mathbb{Z}_2[x]/(x^2+x+1)$

$\mathbb{Z}_2[x]/(x^2+x+1)$ é um corpo finito com 4 elementos. Como veremos na próxima seção, isto não é mera coincidência.

6. Fatoração única de polinômios

Nesta seção provamos o análogo para polinômios do Teorema da Fatoração Única do capítulo 1. Começamos por definir o que corresponde a um número primo, no contexto dos polinômios. Ao longo de toda a seção trabalharemos apenas com polinômios sobre um corpo K.

Seja f um polinômio não constante em $K[x]$. Dizemos que f é *irredutível* em $K[x]$ se não for possível escrevê-lo como produto de dois polinômios de grau menor que o grau de f. Caso contrário, f é um polinômio *redutível*. Observe que ambas as definições dizem respeito, apenas, a polinômios não constantes. Vemos, assim, que as constantes estão para a fatoração de polinômios, como ± 1 estão para a fatoração de inteiros. Para descobrir o que há de comum a estes dois casos resolva os exercícios 7, 8, 9 e 10.

Como consequência imediata da definição, temos que todo polinômio de grau 1 em $K[x]$ é irredutível. Por outro lado, só é possível decompor um polinômio de grau 2 ou 3 como produto de polinômios de grau menor se ao menos um desses polinômios tiver grau 1. Portanto, podemos concluir pelo lema 3.3 que um polinômio de grau 2 ou 3 só poderá ser irredutível em $K[x]$ se não tiver raiz em K. Para polinômios de grau maior, a situação é muito mais complicada. Por exemplo, um polinômio de grau 4 pode ser decomposto como o produto de dois polinômios de grau 2, apesar de não ter nenhuma raiz em K. Finalmente, os possíveis graus de polinômios irredutíveis sobre K dependem do corpo K. Assim, apenas os polinômios de grau 1 são irredutíveis sobre $\mathbb{C}$, ao passo que em $\mathbb{R}[x]$ há polinômios irredutíveis de graus 1 e 2. Já $\mathbb{Q}[x]$ contém polinômios irredutíveis de grau n, para cada inteiro positivo $n \geq 1$; veja **[Gonçalves 1979, p. 84, Exemplo 2]**.

Como ocorre com os números inteiros, é fácil provar que todo polinômio de $K[x]$ pode ser escrito como produto de polinômios irredutíveis. O difícil é mostrar que a fatoração é única. Para isso, precisamos do seguinte lema.

LEMA 3.5. *Sejam p, f e g polinômios em $K[x]$ e suponhamos que p seja irredutível em $K[x]$.*

(i) Se p não divide f, então existem polinômios α e β em $K[x]$ tais que

$$\alpha p + \beta f = 1.$$

(ii) Se p divide fg, então p divide f ou p divide g.

Se estivéssemos trabalhando com inteiros, em vez de polinômios, o item (i) do lema seria uma conseqüência imediata do algoritmo euclidiano estendido, **[Coutinho 2001, p. 44]**. Como há uma versão daquele algoritmo para polinômios, poderíamos ter adotado a mesma estratégia para provar este lema. Só não fizemos isto porque, além da demonstração que daremos a seguir ser bastante curta, não precisamos do algoritmo euclidiano para polinômios em nenhum lugar deste livro.

DEMONSTRAÇÃO. Para provar (i), considere o ideal de $K[x]$ gerado por f e por p. Pelo teorema 3.4, todo ideal de $K[x]$ é principal. Logo deve existir um polinômio h que gera (f, p). Mas, para isto acontecer, h tem que dividir p e f. Contudo, p é irredutível, e só pode ser divisível por h se h for constante, ou se $h = cp$, para algum $c \in K \setminus \{0\}$. Entretanto, neste último caso, p dividiria f, o que estamos supondo que não ocorre. Logo, h deve ser uma constante não nula. Multiplicando h pelo seu inverso em K concluímos que $1 \in (f, p)$. Portanto, pela definição do ideal (f, p) existem polinômios α e β tais que

$$\alpha p + \beta f = 1. \tag{17}$$

Passemos à demonstração de (ii). Se p divide f, então provamos o que queríamos. Suponhamos, então, que p não divide f e vamos mostrar que tem que dividir g. Mas, neste caso, temos por (i) que existem polinômios α e β tais que (17) é satisfeita. Multiplicando (17) por g, obtemos

$$\alpha pg + \beta fg = g. \tag{18}$$

A primeira parcela da esquerda é evidentemente um múltiplo de p; mas a segunda parcela também é, porque p divide fg por hipótese. Portanto, p divide o lado esquerdo da equação (18); logo divide g, o que conclui a demonstração.

Como no caso da fatoração de inteiros, a parte (ii) do lema 3.5 é o ingrediente essencial para a demonstração da unicidade da fatoração polinômios.

TEOREMA 3.6. *Sejam K um corpo e f um polinômio não constante em $K[x]$. Podemos fatorar f na forma*

$$f = cp_1^{e_1} p_2^{e_2} \cdots p_t^{e_t},$$

onde c é uma constante não nula, $p_1, p_2, \ldots, p_t$ são polinômios mônicos irredutíveis distintos e $e_1, \ldots, e_t$ são inteiros positivos. Além do mais, esta fatoração é única a menos da ordenação dos polinômios irredutíveis.

Dizer que a *fatoração é única a menos da ordenação dos polinômios irredutíveis* significa que, se várias pessoas calcularem uma fatoração para f, todas encontrarão a mesma constante, os mesmos polinômios irredutíveis e os

mesmos expoentes para cada polinômio irredutível. O pior que pode acontecer é que pessoas diferentes arranjem as potências de polinômios que aparecem na fatoração em uma ordem diferente.

DEMONSTRAÇÃO. Há duas coisas distintas a provar: a existência e a unicidade da fatoração. Começamos provando a existência da fatoração por indução no grau de f.

EXISTÊNCIA DA FATORAÇÃO: Um polinômio de grau 1 tem que ser irredutível, de modo que já está fatorado. Suponhamos que todo polinômio de grau menor que n admite uma fatoração na forma do teorema. Seja f um polinômio de grau n. Se f for irredutível, já está fatorado e não há nada a fazer. Suponhamos que f não seja irredutível. Neste caso existem polinômios g_1 e g_2, de grau menor que n, tais que $f = g_1 g_2$. Como g_1 e g_2 têm grau menor que n, podemos fatorá-los na forma do teorema. Para obter a fatoração de f basta multiplicar as constantes das fatorações de g_1 e de g_2, e agrupar os fatores irredutíveis iguais nas duas fatorações. Portanto, pelo Princípio de Indução Finita, qualquer polinômio de grau positivo admite uma fatoração em irredutíveis.

UNICIDADE DA FATORAÇÃO: Suponhamos, por contradição, que existe algum polinômio de grau positivo que admite mais de uma fatoração na forma estabelecida pelo teorema. Seja f um polinômio, de menor grau possível, entre aqueles que admitem duas fatorações distintas. Neste caso, podemos escrever

$$f = cp_1^{e_1} \cdots p_t^{e_t} = c' q_1^{r_1} \cdots q_s^{r_s}. \tag{19}$$

onde c, c' são constantes não nulas, $p_1, \ldots, p_t$ e $q_1, \ldots, q_s$ são polinômios mônicos irredutíveis e $e_1, \ldots, e_t, r_1, \ldots, r_s$ são inteiros positivos. Naturalmente, estamos supondo que estas duas fatorações são diferentes.

Em primeiro lugar, como os ps e qs são mônicos, tanto c quanto c' têm que ser iguais ao coeficiente líder de f. Portanto, $c = c'$.

Por outro lado, de acordo com a fatoração da esquerda, p_1 é um polinômio irredutível que divide f. Mas $f = c' q_1^{r_1} \ldots q_s^{r_s}$, segundo a fatoração da direita. Logo, pelo lema 3.5(ii), p_1 deve dividir um dos fatores do produto à direita. Reordenando os qs, se necessário, podemos supor que p_1 divide q_1. Só que, como os dois são irredutíveis e mônicos, um deles só pode dividir o outro se forem iguais.

Com isso podemos reescrever (19) substituindo q_1 por p_1

$$f = cp_1^{e_1} \cdots p_k^{e_t} = c' p_1^{r_1} q_2^{r_2} \ldots q_s^{r_s}.$$

Como temos p_1 em ambos os lados da igualdade, com multiplicidade pelo menos 1, podemos cancelá-lo obtendo

$$g = cp_1^{e_1 - 1} \cdots p_k^{e_t} = c' p_1^{r_1 - 1} \cdots q_s^{r_s}. \tag{20}$$

Mas, $f = p_1 g$, de modo que o grau de f tem que ser maior que o grau de g. Portanto, pela escolha de f, as duas fatorações de g em (20) têm que ser iguais.

Isto significa que, reordenando $q_2, \ldots, q_s$ novamente, se necessário, teremos que $t = s$, e, também que

$$p_i = q_i \text{ e } e_i = r_i \text{ para cada } 1 \leq i \leq t.$$

Mas isto basta para mostrar que as duas fatorações de f dadas anteriormente têm que ser iguais, contradizendo a suposição inicial. Com isto provamos que não pode haver um polinômio de grau positivo com duas fatorações distintas, na forma explicitada pelo teorema.

Ao discutir a fatoração de inteiros no capítulo 1, provamos a existência da fatoração descrevendo um algoritmo que, ao receber um inteiro, encontrava um fator primo. Entretanto, nossa demonstração da existência da fatoração de polinômios, no teorema acima, não se presta a nenhuma formulação algorítmica. Ela mostra que a fatoração existe, mas não dá nenhuma deixa de como possa ser calculada. Isto remete a uma diferença muito interessante entre os anéis $\mathbb{Z}$ e $K[x]$. Se por um lado, é fácil descrever um algoritmo que acha um fator de um número inteiro; ninguém ainda foi capaz de imaginar um algoritmo que faça isto em tempo polinomial. Já para os polinômios, não existe nenhum algoritmo realmente ingênuo de fatoração; em contrapartida, temos neste caso um algoritmo de fatoração de custo polinomial. Uma descrição detalhada deste último algoritmo pode ser encontrada em **[von zur Gathen e Gerhard 1999, capítulo 16]**.

Podemos usar a teorema 3.6 para relacionar o número de raízes de um polinômio com o seu grau, como mostra o próximo corolário.

COROLÁRIO 3.7. *Um polinômio de grau n, com coeficientes em um corpo K, não pode ter mais do que n raízes em K.*

DEMONSTRAÇÃO. Pelo lema 3.3, $\alpha \in K$ é uma raiz de f se, e somente se, $x - \alpha$ divide f. Mas $x - \alpha$ é um polinômio de grau 1, de modo que tem que ser um fator irredutível de f. Contudo, pelo teorema 3.6, a soma dos graus dos fatores irredutíveis de f não pode exceder o grau de f. Em particular, f não pode ter mais fatores lineares (e, portanto, raízes) que o seu grau.

Encerramos este capítulo usando o lema 3.5 para explicar porque o anel quociente do exemplo ao final da seção 4 é um corpo. O corolário seguinte será utilizado de maneira fundamental na demonstração do algoritmo AKS no capítulo 5.

COROLÁRIO 3.8. *Seja K um corpo e p um polinômio irredutível de $K[x]$. Então o anel quociente $K[x]/(p)$ é um corpo. Além do mais, se K for um corpo finito, então $K[x]/(p)$ também será finito.*

DEMONSTRAÇÃO. Precisamos mostrar que toda classe não nula do anel quociente $K[x]/(p)$ é invertível. Mas já vimos na seção anterior que cada uma destas classes admite como representante um polinômio f de grau não

negativo, e menor que o grau de p. Como toda constante não nula é invertível em $K[x]/(p)$, podemos supor que f tem grau positivo. Seja, então,

$$\widehat{f} \in K[x]/(p), \text{ onde } 0 < \operatorname{grau}(f) < \operatorname{grau}(p).$$

Mas f tem grau menor que o grau de p, de modo que não pode ser divisível por p. Logo, pelo lema 3.5(i), existem polinômios α e β tais que

$$\alpha p + \beta f = 1 \text{ onde } \alpha, \beta \in K[x].$$

Tomando esta relação no quociente $K[x]/(p)$, obtemos

$$\widehat{\beta}\widehat{f} = \widehat{1},$$

já que $\widehat{p} = \widehat{0}$. Portanto, $\widehat{\beta}$ é o inverso de $\widehat{f}$ em $K[x]/(p)$.

A afirmação final segue do fato de que se K é finito, então existe apenas uma quantidade finita de polinômios em $K[x]$ cujo grau é menor que o grau de p.

Note que, no exemplo do final da seção 4, o polinômio gerador do ideal que definia o quociente era $p = x^2 + x + \overline{1}$. Mas, para que este polinômio não fosse irredutível em $\mathbb{Z}_2[x]$ seria preciso que pudesse ser decomposto como o produto de dois fatores de grau 1. Pelo corolário 3.7, isto só poderia acontecer se p tivesse raízes em $\mathbb{Z}_2$. Contudo,

$$p(\overline{0}) = \overline{0}^2 + \overline{0} + \overline{1} = \overline{1} \text{ e } p(\overline{1}) = \overline{1}^2 + \overline{1} + \overline{1} = \overline{1}.$$

Logo $p = x^2 + x + \overline{1}$ é irredutível em $\mathbb{Z}_2[x]$. Observe que este argumento não teria funcionado se o polinômio tivesse grau maior que 3. Por exemplo, o argumento acima mostra que $(x^2 + x + 1)^2$ não tem raízes, no entanto, ele não é irredutível em $\mathbb{Z}_2[x]$.

7. Exercícios

1. Mostre que todo domínio com uma quantidade finita de elementos é um corpo.

2. Seja A um anel e $f_1, f_2, g \in A[x]$. Mostre que:
   (a) 0 é divisível por g;
   (b) se g divide f_1 então divide $f_1 f_2$;
   (c) se g divide f_1 e f_2, então divide $f_1 + f_2$.

3. Seja I um ideal de um anel A. Mostre que $I = A$ se, e somente se $1 \in I$.

4. Mostre que se os únicos ideais de um anel A são $\{0\}$ e A, então A é um corpo.
   SUGESTÃO: Se $a \in A$ e $a \neq 0$, então $(a) = A$. Conclua usando o exercício 3.

5. Prove que todos os ideais de $\mathbb{Z}$ são principais.

6. Prove, por contradição, que o ideal $(2, x)$ do anel $\mathbb{Z}[x]$ não é principal.

7. Seja A um anel. Dizemos que um elemento $a \in A$ é *invertível* em A se existe $a' \in A$ tal que $aa' = 1$.
   (a) Mostre que o conjunto $A^\star$ dos elementos invertíveis de A forma um grupo abeliano.
   (b) Determine $\mathbb{Z}^\star$.
   (c) Mostre que $A[x]^\star = A^\star$, para qualquer anel A. Conclua que, se K é um corpo, então os elementos invertíveis de $K[x]$ são as constantes não nulas.

8. Seja A um anel e sejam $a, d \in A \setminus \{0\}$. Dizemos que d é um *divisor* de a em A se existe $c \in A$ tal que $a = dc$. Mostre que:
   (a) Se d é um divisor de a, então também é um divisor de ab, qualquer que seja $b \in A$.
   (b) Se d é um divisor de a e também é um divisor de b então d é um divisor de $a + b$.

9. Seja A um anel e sejam $a, d \in A \setminus \{0\}$. O elemento d é um *divisor próprio* de a se $a = dc$ e se $d, c \notin A^\star$. Suponha que existe uma função $\nu : A \to \mathbb{N}$ tal que:
   - $\nu(ab) = \nu(a)\nu(b)$, quaisquer que sejam $a, b \in A$, e
   - $\nu(a) = 1$ se, e somente se, $a \in A^\star$.

   Prove que d é um divisor próprio de a se, e somente se, $\nu(d) < \nu(a)$.

10. Seja A um anel no qual está definida uma função ν como no exercício anterior. Um elemento $a \in A \setminus A^\star$ é *irredutível* em A se não tem divisores próprios. Prove que todo elemento de A pode ser escrito na forma
$$up_1 \cdots p_t,$$
onde $u \in A^\star$ e $p_1, \ldots, p_t$ são elementos irredutíveis de A.

11. Considere o conjunto
$$\mathbb{Z}[\sqrt{5}] = \{a + b\sqrt{5} : a, b \in \mathbb{Z}\}.$$
   (a) Mostre que $\mathbb{Z}[\sqrt{5}]$ é um domínio.
   (b) Mostre que a função $\nu : \mathbb{Z}[\sqrt{5}] \to \mathbb{N}$ definida por
$$\nu(a + b\sqrt{5}) = |a^2 - 5b^2|$$
   satisfaz as condições do exercício 9.
   (c) Mostre que todo elemento de $\mathbb{Z}[\sqrt{5}]$ pode ser fatorado como produto de elementos irredutíveis.
   (d) Mostre que 2, 3, $1 + \sqrt{5}$ e $1 - \sqrt{5}$ são irredutíveis em $\mathbb{Z}[\sqrt{5}]$.
   (e) Mostre que 6 pode ser fatorado de duas maneiras distintas como produto de irredutíveis em $\mathbb{Z}[\sqrt{5}]$.

12. Use o exercício 4 para provar um análogo do lema 3.5 para o anel $\mathbb{Z}$.

13. Use o exercício anterior para provar a unicidade da fatoração de um inteiro em primos, como estipulada pelo Teorema da Fatoração Única.

14. Seja K um corpo. Prove que um polinômio de grau 2 ou 3 em $K[x]$ é irredutível se, e somente se, não tem raiz em K.

Capítulo 4

Testes de primalidade

Neste capítulo explicamos o que, exatamente, queremos dizer quando nos referimos a um teste de primalidade *determinístico e eficiente*. Faremos isto contrastando testes de primalidade que são determinísticos com testes que não são determinísticos, e analisando a eficiência de cada um deles.

1. Um teste determinístico de primalidade

Antes de tratar dos testes de primalidade, precisamos de alguns resultados básicos da teoria dos números. Ao longo de todo este capítulo x denotará uma variável. Muitos dos cálculos a seguir serão efetuados em $\mathbb{Z}_n[x]$, o anel de polinômios com coeficientes em $\mathbb{Z}_n$.

TEOREMA 4.1. *Sejam n e b inteiros positivos e primos entre si. Então,*

$$(x+\overline{b})^n = x^n + \overline{b}^n \quad \textit{em} \quad \mathbb{Z}_n[x],$$

se, e somente se, n é primo.

A demonstração deste teorema depende de um resultado sobre números binomiais que isolaremos no seguinte lema.

LEMA 4.2. *Seja n um inteiro positivo. Então, n é primo se, e somente se,*

$$\binom{n}{k} \equiv 0 \pmod{n}$$

para todo $1 \leq k \leq n-1$.

Observe que, como

$$\binom{n}{0} = \binom{n}{n} = 1,$$

não podemos estender o intervalo em que k está variando de modo que inclua 0 e n.

DEMONSTRAÇÃO. Suponha, em primeiro lugar, que $n = p$ é um número primo. Neste caso, se $1 \leq k \leq p-1$, então p não divide $k!$ nem $(p-k)!$. Contudo p divide $p!$. Portanto, o p que aparece no numerador do quociente

$$\binom{p}{k} = \frac{p!}{k!(p-k)!}$$

não é cancelado pelo denominador. Mas isto significa que

$$\binom{p}{k} \text{ é múltiplo de } p, \text{ quando } 1 \leq k \leq p-1$$

como queríamos mostrar.

Vejamos o que acontece se n for composto. Neste caso, n admite um fator primo próprio q. Em outras palavras, q divide n e $1 < q \leq n-1$. Digamos que q^e é a maior potência de q que divide n. Assim, podemos escrever $n = q^e c$, onde c é um inteiro positivo que não é múltiplo de q. Temos, então, que

$$\binom{n}{q} = \frac{n \cdot (n-1)!}{q!(n-q)!} = q^{e-1} c \cdot \frac{(n-1)!}{(q-1)!(n-q)!}.$$

Precisamos mostrar que este número não é múltiplo de n. Mas, se fosse, então q dividiria $(n-1)!/(q-1)!(n-q)!$ que, por sua vez, é igual a

$$\frac{(n-1)(n-2)\cdots(n-q+1)(n-q)!}{(q-1)!(n-q)!} = \frac{(n-1)(n-2)\cdots(n-q+1)}{(q-1)!}.$$

No entanto, como n é múltiplo de q, os números $(n-1), (n-2), \ldots, (n-q+1)$ não podem ser múltiplos de q. Disto segue que $(n-1)!/(q-1)!(n-q)!$ não é divisível por q, completando assim a demonstração do lema.

Estamos, agora, prontos para provar o teorema.

DEMONSTRAÇÃO DO TEOREMA 4.1 Em primeiro lugar, temos, pelo binômio de Newton, que

$$(x+b)^n = x^n + b^n + \sum_{k=1}^{n-1} \binom{n}{k} x^{n-k} b^k.$$

Portanto, o teorema estará provado se mostrarmos que

$$\sum_{k=1}^{n-1} \binom{n}{k} x^{n-k} \overline{b}^k = \overline{0} \text{ em } \mathbb{Z}_n[x], \tag{21}$$

se, e somente se, n é primo. Entretanto, como $\mathrm{mdc}(b, n) = 1$, temos pelo teorema de inversão de **[Coutinho 2001, p. 83]**, que

$$\binom{n}{k} b^k \equiv 0 \pmod{n} \text{ se, e somente se, } \binom{n}{k} \equiv 0 \pmod{n}.$$

Mas esta última congruência é uma conseqüência imediata do lema 4.2, já que $1 \leq k \leq n-1$. Portanto, cada parcela em (21) é congruente a 0 módulo n, o que prova o teorema.

Antes de prosseguir, provaremos uma generalização de uma das metades do teorema 4.1. Esta proposição será utilizada na demonstração do algoritmo AKS, no próximo capítulo.

PROPOSIÇÃO 4.3. *Sejam $p > 0$ um primo e b e $k \geq 0$ inteiros. Então,*

$$(x + \overline{b})^{p^k} = x^{p^k} + \overline{b}^{p^k} \text{ em } \mathbb{Z}_p[x].$$

Note que, ao contrário do que aconteceu no enunciado do teorema 4.1, não estamos supondo que $\mathrm{mdc}(p, b) = 1$.

DEMONSTRAÇÃO. A demonstração será por indução em k. Se $k = 1$, o resultado é conseqüência imediata do binômio de Newton e do lema 4.2, à maneira do que fizemos na demonstração do teorema 4.1; veja exercício 2.

Todas as igualdades de polinômios nesta demonstração ocorrem no anel $\mathbb{Z}_p[x]$. Suponha, então, que

$$(x + \overline{b})^{p^k} = x^{p^k} + \overline{b}^{p^k}$$

para algum inteiro $k \geq 1$, e vamos provar que uma congruência semelhante vale quando substituímos k por $k + 1$. Mas,

$$(x + \overline{b})^{p^{k+1}} = ((x + \overline{b})^{p^k})^p = (x^{p^k} + \overline{b}^{p^k})^p, \tag{22}$$

pela hipótese de indução. Entretanto, pelo teorema 4.1

$$(x + \overline{b}^{p^k})^p = x^p + \overline{b}^{p^{k+1}}.$$

Como x é uma variável, podemos substituí-la por qualquer valor que desejarmos. Trocando, então, x por x^{p^k}, obtemos

$$(x^{p^k} + \overline{b}^{p^k})^p = x^{p^{k+1}} + \overline{b}^{p^{k+1}}. \tag{23}$$

Finalmente, combinando (22) com (23), concluímos que

$$(x + \overline{b})^{p^{k+1}} = (x^{p^k} + \overline{b}^{p^k})^p = x^{p^{k+1}} + \overline{b}^{p^{k+1}},$$

como queríamos mostrar.

Vejamos como o teorema 4.1 pode ser usado para dar uma demonstração do teorema de Fermat, ao qual está intimamente relacionado.

TEOREMA DE FERMAT. *Seja p um primo positivo e b um número inteiro. Se p não divide b então*

$$b^{p-1} \equiv 1 \pmod{p}.$$

DEMONSTRAÇÃO. Como $\mathrm{mdc}(b, p) = 1$ então b é inversível módulo p. Logo,

$$b^{p-1} \equiv 1 \pmod{p} \text{ se, e somente se, } b^p \equiv b \pmod{p}.$$

Provaremos esta segunda congruência, para todo inteiro *positivo* b, usando o teorema 4.1 e indução em b. Isto é suficiente para provar o resultado geral porque todo inteiro é congruente a seu resíduo módulo p que, por sua vez, é um número não negativo.

O resultado é evidentemente verdadeiro se $b = 0$ ou $b = 1$. Suponhamos que $b^p \equiv b \pmod{p}$, e vamos mostrar que $(b + 1)^p \equiv b + 1 \pmod{p}$. Mas, de acordo com o teorema 4.1,

$$(x + \overline{1})^p = x^p + \overline{1}^p = x^p + \overline{1} \text{ em } \mathbb{Z}_p[x].$$

Substituindo x por b, obtemos

$$(b+1)^p \equiv b^p + 1 \pmod{p}.$$

Entretanto, $b^p \equiv b \pmod{p}$, pela hipótese de indução. Portanto,

$$(b+1)^p \equiv b^p + 1 \equiv b + 1 \pmod{p},$$

como queríamos mostrar.

Combinando o teorema de Fermat com o teorema 4.1, obtemos o seguinte resultado, que desempenhará um papel chave na nossa investigação.

COROLÁRIO 4.4. *Sejam n e b inteiros positivos primos entre si. Então,*

$$(x+\overline{b})^n = x^n + \overline{b} \text{ em } \mathbb{Z}_n[x]$$

se, e somente se, n é primo.

Note que a diferença entre o teorema 4.1 e o corolário 4.4 é que o b^n do lado direito da congruência passou a ser b no corolário. Podemos fazer esta troca, quando n é primo, por conta do teorema de Fermat. Por outro lado, se n não for primo, a troca não afeta o fato de haver números binomiais que não se anulam módulo n.

A expressão *teste de primalidade* refere-se a um algoritmo que, tendo por entrada um inteiro positivo n, determina se n é, ou não é, primo. Para obter um tal teste, precisamos de alguma propriedade que seja verdadeira se, e somente se, n for primo, e é exatamente isto que o corolário 4.4 nos dá. Portanto, usando o corolário, podemos formular o seguinte teste.

Teste de primalidade determinístico

Entrada: inteiro positivo n.

Saída: mensagem indicando se n é ou não primo.

Primeira etapa: Calcule todos os termos da expansão $(x+\overline{b})^n$ em $\mathbb{Z}_n[x]$.

Segunda etapa: Verifique se a expressão obtida na etapa anterior é igual a $x^n + \overline{b}$. Se for n é primo, se não for n é composto.

Isto é o que se chama de um *teste determinístico de primalidade*. Ele é *determinístico* porque, seja a resposta primo ou composto, temos a garantia de que está correta. Mas, nem todos os testes de primalidade são determinísticos. Alguns garantem apenas que o número é primo com uma certa probabilidade, que pode ser controlada de acordo com a necessidade do usuário. Na verdade, a maioria dos testes de primalidade usados na prática são deste último tipo. Mas, por que alguém usaria um teste não determinístico, quando já temos um que é determinístico? A resposta está na eficiência deste teste.

Já que determinar o custo de execução do teste acima exige um certo cuidado, vamos nos contentar com uma avaliação muito grosseira. Observe que, na primeira etapa, temos que calcular todos os termos de $(x+\overline{b})^n$. Como esta expressão binomial tem $n+1$ termos, o número de operações aritméticas a ser

efetuado não pode ser menor que n. À primeira vista isto pode não parecer ruim. Só que, anteriormente, vínhamos expressando o custo dos algoritmos em função, não da entrada, mas sim do número de bits da entrada. Mas se n tiver $k+1$ bits, então $n \geq 2^k$. Assim, o custo deste teste de primalidade não pode ser melhor do que $O(2^k)$. Portanto, trata-se de um teste cujo tempo de execução é exponencial, o que explica porque não é eficiente.

2. Testes Não Determinísticos de Primalidade

Muitos dos testes não determinísticos de primalidade utilizados na prática estão baseados no teorema de Fermat. A idéia é muito simples. Digamos que queremos saber se um dado inteiro $n > 0$ é ou não primo. De acordo com o teorema de Fermat, se n for primo então para todo inteiro $0 < b < n$ temos que $b^{n-1} \equiv 1 \pmod{n}$. Imagine agora que encontramos um inteiro b, entre 1 e $n-1$, para o qual $b^{n-1} \not\equiv 1 \pmod{n}$. Então, o teorema de Fermat nos garante que n tem que ser composto.

Composto? Mas, e se queremos mostrar que n é primo? Num momento de entusiasmo poderíamos imaginar que a recíproca do teorema de Fermat deve ser verdadeira; ou seja, que

$$\text{se} \quad b^{n-1} \equiv 1 \pmod{n} \text{ então } n \text{ é primo.}$$

E estaríamos em boa companhia ao pensar isto, afinal Leibniz achava o mesmo. Infelizmente, nem por isso o resultado deixa de ser falso. E muito falso! De fato, existem números compostos, conhecidos como *números de Carmichael*, para os quais

$$b^{n-1} \equiv 1 \pmod{n}$$

para todo $1 \leq b \leq n-1$ que satisfaz $\mathrm{mdc}(b,n) = 1$. O menor número de Carmichael é 561, mas foi demonstrado em 1994 que existe uma infinidade destes números; veja **[Alford, Granville e Pomerance 1994]** para mais detalhes. Assim, se n for um número de Carmichael cujo menor fator é muito grande, será praticamente impossível detectar isto calculando b^{n-1} módulo n, mesmo que usemos muitos valores diferentes de b.

Entretanto, nem tudo está perdido. Na verdade, não é difícil modificar a idéia de usar a recíproca do teorema de Fermat de modo a obter um teste de primalidade muito melhor. Em linhas gerais o que faremos é determinar uma propriedade que todo primo deve satisfazer e que é mais forte que o teorema de Fermat. Se um número inteiro não satisfaz esta propriedade, então tem que ser composto.

Suponhamos que $n > 0$ é um número *ímpar*. Para poder aplicar o teste, escolheremos um inteiro $1 < b < n-1$, conhecido como a *base* do teste. Como n é ímpar, $n-1$ será par, e podemos escrevê-lo na forma $n-1 = 2^k q$, onde q é um inteiro ímpar e $k \geq 1$. Isto é, fatoramos a maior potência de 2 possível em $n-1$. A seguir, calculamos a seguinte seqüência de potências *módulo* n,

$$b^q,\ b^{2q}, \ldots, b^{2^{k-1}q}, b^{2^k q}.$$

Vejamos o que acontece a esta seqüência se n for *primo*. Em primeiro lugar, pelo menos uma destas potências tem que ser congruente a 1 módulo n porque, pelo teorema de Fermat

$$b^{2^k q} \equiv b^{n-1} \equiv 1 \pmod{n}.$$

Digamos que j é o *menor* expoente tal que $b^{2^j q} \equiv 1 \pmod{n}$. Se $j \geq 1$, podemos usar produtos notáveis para escrever

$$b^{2^j q} - 1 = (b^{2^{j-1} q} - 1)(b^{2^{j-1} q} + 1).$$

Como estamos supondo que n é primo e divide $b^{2^j q} - 1$, então, n divide $b^{2^{j-1} q} - 1$, ou n divide $b^{2^{j-1} q} + 1$. Mas j é o *menor* expoente tal que $b^{2^j q} - 1$ é divisível por n. Portanto, $b^{2^{j-1} q} - 1$ não pode ter n como fator. Logo $b^{2^{j-1} q} + 1$ tem que ser divisível por n; isto é, $b^{2^{j-1} q} \equiv -1 \pmod{n}$. Note, entretanto, que estes cálculos dependem do fato de que $j \geq 1$ e, portanto, não podem ser efetuados se $b^q \equiv 1 \pmod{n}$. Concluímos, assim, que:

Se n é primo então, ou a seqüência

$$b^q, b^{2q}, \ldots b^{2^{k-1} q} \tag{24}$$

módulo n, começa com 1, ou tem -1 em alguma de suas posições (incluindo a primeira).

Portanto, se nenhuma destas duas coisas é verificada para n, com respeito a algum b entre 1 e $n-1$, então podemos estar seguros de que n é composto.

Analisando em detalhe o argumento utilizado para chegar à conclusão do parágrafo acima, verificamos que a primalidade de n foi usada em dois pontos cruciais. Em primeiro lugar, precisamos que n fosse primo para garantir a existência de um 1 em alguma posição da seqüência. Em segundo lugar, usamos que n é primo para assegurar que, se o produto notável fosse congruente a 0 módulo n, o mesmo ocorreria com um dos seus fatores.

O problema é que, mais uma vez, temos um teste que apenas detecta se o número é composto. Por exemplo, calculando a seqüência (24) com $n = 2047$ e $b = 2$, obtemos somente o termo

$$2^{1023} \equiv 1 \pmod{2047},$$

pois $n-1 = 2046 = 2 \cdot 1023$. Logo, o número 2047 comporta-se como se fosse um número primo, com respeito à condição explicitada acima. Entretanto, $2047 = 23 \cdot 89$, é um número composto.

O algoritmo que implementa este teste é o seguinte.

Teste forte de composição

Entrada: um inteiro ímpar n, que se quer testar, e a base b, onde $1 < b < n-1$.

Saída: uma das mensagens: *'n é composto'* ou *'teste inconclusivo'*.

Etapa 1: Divida $n-1$ sucessivamente por 2 até encontrar q (um número ímpar) e k tais que $n - 1 = 2^k q$.

Etapa 2: Comece fazendo $i = 0$ e $r =$ resíduo de b^q por n.

Etapa 3: Se $i = 0$ e $r = 1$ ou se $i \geq 0$ e $r = n - 1$ a saída é *'teste inconclusivo'*.

Etapa 4: Incremente i de 1 e substitua r pelo resto da divisão de r^2 por n.

Etapa 5: Se $i < k - 1$ volte à etapa 3, senão a saída é *'n é composto'*.

O nome que demos ao teste merece uma explicação. Em primeiro lugar, preferimos chamá-lo teste de *composição*, em vez de teste de *primalidade*, porque já sabemos que a única coisa que ele detecta com segurança é que um número é composto. Em segundo lugar, o adjetivo *forte* foi acrescentado para distingui-lo do teste do início da seção. Afinal, aquele teste é mais fraco que este, pois verifica apenas a congruência de Fermat. Pelo resto da seção vamos nos referir ao teste acima, abreviadamente, como *teste forte*.

O teste forte foi proposto originalmente por M. Artjuhov em um artigo (em russo) publicado em 1967. Redescoberto mais tarde por J. Selfridge, foi popularizado através dos trabalhos de M. Rabin e de G. L. Miller em 1976. O que eles propuseram, foram maneiras de tornar o teste mais efetivo, aplicando-o a várias bases diferentes. Para entender melhor como isto é possível, vamos comparar o teste forte ao teste mais ingênuo do início desta seção. Para começar, não existe nada semelhante a um número de Carmichael relativamente ao teste forte. Mais precisamente, se o teste forte tiver saída inconclusiva quando aplicado a mais de $n/4$ bases no intervalo que vai de 1 a $n - 1$, então n será necessariamente primo. Uma demonstração deste fato pode ser encontrada em **[Crandall e Pomerance 2002, seção 3.4]** e também em **[Monier 1980]** e **[Rabin 1980]**, onde o resultado foi originalmente provado. Talvez você esteja pensando:

> "grande coisa, se n for da ordem de 10^{200} isto significaria aplicar o teste a $10^{200}/4$ bases, o que é evidentemente impossível!"

Apesar disto ser verdade, há uma maneira muito mais satisfatória de interpretar esta afirmação. Suponha, como antes, que n é um inteiro ímpar composto. Note que, ao escolher b aleatoriamente, há uma probabilidade de apenas $1/4$ de que b seja escolhido de modo que n teste inconclusivo com respeito ao teste forte. Portanto, se um dado n testar inconclusivo para uma base b então a probabilidade de que seja composto é de $1/4$. Entretanto, se testarmos duas bases, e se n testar inconclusivo para ambas, então a probabilidade cai para $(1/4)^2 = 1/16$. Continuando assim vemos que, se n testar inconclusivo para 10 bases distintas, então a probabilidade de que seja composto caiu para $(1/4)^{10}$, que é da ordem de 10^{-6}. É claro que um tal número é quase que certamente primo. Esta é a estratégia adotada para testar primalidade em quase todos os sistemas de computação algébrica.

Seja como for, ainda há um 'quase' presente, e que é muito significativo: não importa quantas bases escolhermos, sempre haverá um número composto

n que testará inconclusivo para o teste forte com respeito a todas estas bases. Naturalmente, quanto maior for o número de bases, maior será n.

3. Exponenciação

Até aqui, descrevemos um teste probabilístico de primalidade e discutimos suas chances de determinar, efetivamente, que um dado inteiro ímpar é primo. Entretanto, a afirmação feita no final da seção 1 ia além disto. De fato, dissemos também que um tal teste teria custo polinomial, ao contrário do teste determinístico cujo custo é exponencial. Para verificar a veracidade desta afirmação precisamos analisar a eficiência do teste forte.

Em primeiro lugar, o teste precisa descobrir qual é a maior potência de 2 que divide $n-1$. Como $n-1$ está representado no computador em base 2, isto tem custo praticamente zero. O que realmente importa, portanto, é o custo de calcular a seqüência (24). Mas se $n-1 = 2^k q$, onde q é ímpar, estaremos calculando, no máximo, k potências de b. Como $k \leq \log_2 n$, o teste será polinomial desde que possamos calcular cada potência a um custo polinomial. O problema é que a maneira usual de calcular b^q é usar o fato de que $b^q = b^{q-1} \cdot b$. Em outras palavras, para chegar a b^q, o que calculamos de fato são os números

$$\begin{array}{ccccc} b \cdot b, & b^2 \cdot b, & b^3 \cdot b, & \ldots, & b^{q-1} \cdot b \\ \shortparallel & \shortparallel & \shortparallel & & \shortparallel \\ b^2 & b^3 & b^4 & & b^q \end{array}$$

Contudo, efetuamos $q-1$ multiplicações para calcular esta seqüência, de modo que este algoritmo terá um custo exponencial.

Por sorte, o problema se restringe a uma escolha inadequada de algoritmo. Para solucioná-lo basta utilizar um procedimento que calcula a potência "um quadrado de cada vez". O curioso é que os Egípcios do tempo dos Faraós utilizavam um método semelhante para calcular os múltiplos de um número **[Ritter 1997, p. 70]**.

Para entender melhor a estratégia deste novo algoritmo, digamos que quiséssemos calcular b^{2^r}. Enquanto o procedimento anterior baseava-se no fato de que $b^{2^r} = b \cdot b^{2^r-1}$, nossa estratégia agora será baseada na fórmula $b^{2^r} = (b^{2^{r-1}})^2$. Portanto, obtemos b^{2^r} calculando a seqüência

$$\begin{array}{ccccc} b \cdot b, & b^2 \cdot b^2, & b^4 \cdot b^4, & \ldots, & b^{2^{r-1}} \cdot b^{2^{r-1}} \\ \shortparallel & \shortparallel & \shortparallel & & \shortparallel \\ b^2 & b^4 & b^8 & & b^{2^r} \end{array}$$

Neste caso, efetuamos apenas r quadrados, de modo que o número de multiplicações necessárias é polinomial.

Vejamos, em mais detalhes, como o algoritmo funciona. Suponhamos que são dados três inteiros positivos a, e e n. O algoritmo deverá calcular o resíduo de a^e módulo n, usando a expansão binária do expoente. Digamos que $e = (b_m, \ldots, b_0)_2$. Como

$$e = b_m 2^m + \cdots + b_1 2 + b_0,$$

teremos que

$$a^e = (a^2)^{b_m 2^{m-1} + \cdots + b_2 2 + b_1} \cdot a^{b_0}.$$

Observe que a^{b_0} pode valer 1 (se $b_0 = 0$) ou a (se $b_0 = 1$). Seja $P_1 = a^{b_0}$; continuando, temos

$$a^e = (a^4)^{b_m 2^{m-2} + \cdots + b_3 2 + b_2} \cdot (a^2)^{b_1} P_1.$$

Seja $P_2 = (a^2)^{b_1} P_1$; temos então

$$a^e = (a^8)^{b_m 2^{m-3} + \cdots + b_4 2 + b_3} \cdot (a^4)^{b_2} P_2.$$

Continuando assim, teremos uma seqüência de inteiros $P_1, P_2, \ldots P_m$ onde $P_m = a^e$. Naturalmente se estamos calculando em $\mathbb{Z}_n$, efetuaremos a redução módulo n de cada uma das multiplicações que aparecem no algoritmo. Assim, em cada etapa, há apenas duas multiplicações básicas:

(i) o cálculo de um quadrado para obter a seqüência de potências;

(ii) o produto $a^{2^i} P_i$, para $i = 1, 2, \ldots, n$.

Além disso, se na etapa i temos $b_i = 0$, então não é necessário efetuar o produto $a^{2^i} P_i$. Isto nos permite determinar aproximadamente o número de multiplicações necessárias para calcular a^e. Por (i) o número de quadrados que precisamos calcular é aproximadamente $\log_2 e$; ao passo que, por (ii) há tantas multiplicações quanto há algarismos 1 em e. Denotando por $w(e)$ este último número, concluímos que o número de multiplicações é aproximadamente $\log_2 e + w(e) \leq 2 \log_2 e + 1$.

Na prática o algoritmo determina a expansão binária de e à medida que calcula a potência. De fato, se e é ímpar então $b_0 = 1$; se par, então $b_0 = 0$. Podemos determinar b_1 de maneira semelhante, analisando a paridade de

$$b_m 2^{m-1} + \cdots + b_2 2 + b_1.$$

Observe que este último número é $e/2$, se e é par, e $(e-1)/2$ se e é ímpar; e assim por diante. O algoritmo é o seguinte:

Algoritmo de exponenciação

Entrada: Inteiros a, e e n, onde $a, n > 0$ e $e \geq 0$.

Saída: a forma reduzida de a^e módulo n.

Etapa 1: Comece com $A = a$, $P = 1$ e $E = e$;

Etapa 2: Se $E = 0$ imprima P e páre; senão vá para a etapa 3;

Etapa 3: Se E for ímpar então atribua a P o valor do resto da divisão de $A \cdot P$ por n e a E o valor $(E-1)/2$ e vá para etapa 5, senão vá para a etapa 4;

Etapa 4: Se E for par então atribua a E o valor $E/2$ e vá para a etapa 5;

Etapa 5: Substitua o valor atual de A pelo resto da divisão de A^2 por n e vá para a etapa 2.

Já sabemos que o número de multiplicações que o algoritmo precisa efetuar é aproximadamente igual a $2\log_2 e$. Entretanto, isto não representa o custo real do algoritmo, porque ainda precisamos levar em conta o custo de cada multiplicação módulo n. Mas, a multiplicação de duas classes $\overline{a}$ e $\overline{b}$ em $\mathbb{Z}_n$ é feita em duas etapas. Primeiro, efetuamos a multiplicação de inteiros ab; depois, dividimos ab por n e tomamos o resto. Como $a, b < n$, o custo de multiplicá-los será $O((\log_2 n)^2)$, e o custo de dividir $ab < n^2$ por n também será $O((\log_2 n)^2)$. Levando em conta que efetuamos $2\log_2 e$ multiplicações módulo n para calcular a^e módulo n, concluímos que o custo total será da ordem de $O((\log_2 e)(\log_2 n)^2)$.

Finalmente, retornando ao teste forte, temos que o exponente de cada potência não pode exceder $n-1$. Portanto, neste caso, $e < n$ e assim, o custo de calcular b^q é da ordem de $O((\log_2 n)^3)$. Para determinar o resto da seqüência precisamos ainda calcular alguns quadrados. Mas não pode haver mais do que $\log_2 n$ quadrados, e o custo de calcular cada um é $O((\log_2 n)^2)$ pela discussão anterior. Deste modo, o custo total de aplicar o teste forte é de $O((\log_2 n)^3)$, de modo que o teste é realmente polinomial.

Ainda há um ponto importante, para o qual precisamos chamar a sua atenção. Embora o algoritmo de exponenciação descrito acima possa ser utilizado para calcular potências *em qualquer anel*, nem sempre ele terá uma performance polinomial. Isto ocorre, por exemplo, se o algoritmo for utilizado para potências de inteiros ou polinômios. Para entender melhor qual é o problema, digamos que desejamos calcular a^e, onde $a = e = 2^r$. Para isso, teremos que efetuar r multiplicações. Contudo, o custo destas multiplicações não será fixo. Por exemplo, como $a = 2^r$ é um inteiro com $r+1$ bits, então $a^2 = 2^{2r}$ terá $2r+1$ bits. Logo, se o custo de calcular a^2 é $O(r^2)$, o custo de calcular $(a^2)^2$ será $O(4r^2)$. Portanto, o custo total para calcular a^e neste exemplo será da ordem de

$$r^2 + (2r)^2 + (4r)^2 + \cdots + (2^{r-1}r)^2,$$

ou seja $O(2^r)$ e, portanto, exponencial. Uma análise semelhante se aplica à exponenciação de polinômios, mesmo que os coeficientes estejam em $\mathbb{Z}_n$. Portanto, o custo do teste determinístico da seção 1 não vai deixar de ser exponencial, mesmo se usarmos o algoritmo de exponenciação desta seção para calcular $(x+\overline{b})^n$.

4. Balanço

Neste capítulo estudamos dois testes de primalidade representativos da situação anterior ao aparecimento do AKS. Estes testes se dividem em duas categorias. De um lado estão os testes determinísticos, como o que foi apresentado na seção 1. Tendo como entrada um inteiro ímpar n, estes testes certificam se n

é primo ou composto. O problema é que têm uma performance que se torna rapidamente muito ruim à medida que n cresce. Mais precisamente, têm custo exponencial.

Do outro lado estão os testes probabilísticos, como o teste forte de composição. Neste caso, o problema não é a performance, já que têm um custo polinomial, mas sim o fato de que o teste apenas garante que o número é primo dentro de uma certa margem de erro.

A beleza do AKS é que reúne o melhor desses dois mundos: apesar de ser um teste determinístico, tem um custo polinomial.

Entretanto, a análise deste capítulo não esgota todos os tipos de teste de primalidade existentes antes do surgimento do AKS. Do que dissemos pode até ficar a impressão de que seria impossível, até uns anos atrás, determinar com segurança que um número grande era primo – o que, definitivamente, não era verdade. O fato é que existe uma categoria de testes de primalidade que não abordamos anteriormente, e que são uma espécie de simétrico dos que foram discutidos na seção 2. De fato, ao passo que só podemos ter confiança absoluta no teste forte quando retorna que o número é composto, há testes de primalidade em que só podemos confiar se retornam que o número é primo. A esta categoria pertencem os testes descobertos de E. Lucas, e mencionados na introdução. Mais detalhes podem ser encontrados nos capítulos 9 e 10 de **[Coutinho 2001]**.

5. Exercícios

1. O objetivo desta questão é dar uma outra demonstração do teorema de Fermat. Seja $p \geq 2$ um primo e b um inteiro que não é divisível por p. Então $\overline{b} \in U(p)$.
   (a) Mostre que o conjunto
   $$\{\overline{b} \cdot \overline{1}, \overline{b} \cdot \overline{2}, \ldots, \overline{b} \cdot \overline{p-2}, \overline{b} \cdot \overline{p-1}\},$$
   é igual a $U(p)$.
   (b) Mostre que
   $$(\overline{b} \cdot \overline{1}) \cdot (\overline{b} \cdot \overline{2}) \cdots (\overline{b} \cdot \overline{p-2}) \cdot (\overline{b} \cdot \overline{p-1}) = \overline{(p-1)!}.$$
   (c) Cancelando $\overline{(p-1)!}$ conclua que $b^{p-1} \equiv 1 \pmod{p}$.

2. Seja $p \geq 2$ um primo. Prove que
   $$(x + \overline{b})^p = x^p + \overline{b}^p \text{ em } \mathbb{Z}_p[x],$$
   qualquer que seja o inteiro b.

3. Prove que, se $n > 1$ e $\mathrm{mdc}(b^n - b, n) = 1$ para algum inteiro b, então n não pode ser uma potência de primo.

4. Um número ímpar composto n que satisfaz $b^{n-1} \equiv 1 \pmod{n}$ é chamado de *pseudoprimo para a base* b. Mostre que 341, 1729 e 2047 são pseudoprimos para a base 2. Quais destes números também são pseudoprimos para a base 3? E para a base 5?

5. Um número ímpar composto n que testa inconclusivo para o teste forte com respeito a uma base $b > 0$ é chamado de *pseudoprimo forte para a base* b. Mostre que 2047 e 1373653 são pseudoprimos fortes para a base 2. Quais destes números também são pseudoprimos para a base 3? E para a base 5?

6. Prove que todo pseudoprimo forte para a base b é um pseudoprimo para esta mesma base. Dê um exemplo para mostrar que a recíproca desta afirmação é falsa.

7. Seja $n > 2$. Mostre que se $b^{n-1} \equiv 1 \pmod{n}$ para todo inteiro b que é relativamente primo a n, então n é ímpar.

8. Seja n um número composto ímpar. Mostre que o conjunto das bases $1 \leq b < n$ com relação às quais n é um pseudoprimo forma um subgrupo de $U(n)$.

9. Mostre que todo número de Fermat $F(n) = 2^{2^n} + 1$ composto é um pseudoprimo para a base 2.

10. Seja $p > 5$ um primo e considere o número $n = (4^p + 1)/5$.
    (a) Mostre que n é inteiro.
    (b) Mostre que
    $$4^p + 1 = (2^p - 2^{(p+1)/2} + 1)(2^p + 2^{(p+1)/2} + 1),$$
    e conclua que n é composto.
    (c) Mostre que $n - 1 = 2^2 b$, onde b é um número ímpar.
    (d) Use o teorema de Fermat para mostrar que b é divisível por p.
    (e) Prove que $2^{2b} \equiv -1 \pmod{n}$.
    (f) Mostre que n é um pseudoprimo forte para a base 2.

11. Seja $p > 5$ um primo. O número $n = (4^p + 1)/5$ é necessariamente um pseudoprimo forte para a base 3?

12. Seja n um inteiro positivo.
    (a) Mostre que $\overline{n-1}$ tem sempre ordem 2 em $U(n)$.
    (b) Mostre que, se p é primo, então o único elemento de ordem 2 de $U(p)$ é $\overline{n-1}$.
    (c) Mostre que, se p é primo, então o produto de todos os elementos de $U(p)$ é igual a $\overline{n-1}$.
    (d) Mostre que, se n é composto, então $(n-1)! \equiv 0 \pmod{n}$.

(e) Prove o teorema de Wilson, segundo o qual

$$(n-1)! \equiv -1 \pmod{n}$$

se, e somente se, n é primo.

13. Mostre como usar o teorema de Wilson para descrever um teste determinístico de primalidade. Analise a eficiência deste teste, e mostre que é exponencial.

Capítulo 5

O algoritmo AKS

Finalmente é chegada a hora de descrever o algoritmo AKS, de provar que funciona corretamente e que é um algoritmo polinomial.

1. O teorema fundamental

Nesta seção provamos o teorema sobre o qual está baseado o algoritmo AKS. Antes, porém, precisamos isolar um lema que será usado na demonstração do teorema.

LEMA 5.1. *Sejam n, r e b inteiros positivos e $p > 1$ um número primo. Suponha que*

$$(\hat{x} + \hat{b})^n = \hat{x}^n + \hat{b} \text{ em } \mathbb{Z}_p[x]/(x^r - 1).$$

Então

$$(\hat{x} + \hat{b})^{n^i p^j} = \hat{x}^{n^i p^j} + \hat{b} \text{ em } \mathbb{Z}_p[x]/(x^r - 1),$$

quaisquer que sejam os inteiros $i, j \geq 0$.

DEMONSTRAÇÃO. Em primeiro lugar provaremos, por indução em i, que

$$(\hat{x} + \hat{b})^{n^i} = \hat{x}^{n^i} + \hat{b} \text{ em } \mathbb{Z}_p[x]/(x^r - 1), \tag{25}$$

qualquer que seja $i \geq 0$.

O resultado é imediato quando $i = 0$, e coincide com a hipótese do lema quando $i = 1$. Suponhamos, então, que a congruência (25) vale para algum $i \geq 0$ e vamos provar a congruência análoga para $i + 1$. Da hipótese do lema, segue que

$$(x + b)^n = x^n + b + q(x)(x^r - 1),$$

para algum $q(x) \in \mathbb{Z}_p[x]$. Substituindo x por x^{n^i} nesta equação, obtemos

$$(x^{n^i} + b)^n = x^{n^{i+1}} + b + q(x^{n^i})(x^{n^i r} - 1).$$

Mas,

$$\hat{x}^{n^i r} - \hat{1} = (\hat{x}^r)^{n^i} - \hat{1} = \hat{1} - \hat{1} = \hat{0} \text{ em } \mathbb{Z}_p/(x^r - 1),$$

de modo que

$$(\hat{x}^{n^i} + \hat{b})^n = \hat{x}^{n^{i+1}} + \hat{b} \text{ em } \mathbb{Z}_p[x]/(x^r - 1).$$

Entretanto, pela hipótese de indução,

$$(\hat{x} + \hat{b})^{n^{i+1}} = ((\hat{x} + \hat{b})^{n^i})^n = (\hat{x}^{n^i} + \hat{b})^n$$

em $\mathbb{Z}_p[x]/(x^r-1)$. Portanto,

$$(\hat{x}+\hat{b})^{n^{i+1}} = \hat{x}^{n^{i+1}} + \hat{b} \text{ em } \mathbb{Z}_p[x]/(x^r-1),$$

o que prova (25).

Por outro lado, pela proposição 4.3 da página 62,

$$(\hat{x}^{n^i}+\hat{b})^{p^j} = \hat{x}^{n^i p^j} + \hat{b}^{p^j} \text{ em } \mathbb{Z}_p[x].$$

Combinando esta igualdade com o teorema de Fermat, obtemos

$$(\hat{x}^{n^i}+\hat{b})^{p^j} = \hat{x}^{n^i p^j} + \hat{b} \text{ em } \mathbb{Z}_p[x].$$

Contudo, qualquer igualdade em $\mathbb{Z}_p[x]$ dá lugar a uma igualdade no quociente $\mathbb{Z}_p[x]/(x^r-1)$. Assim, segue da equação (25) e do que acabamos de fazer que

$$(\hat{x}+\hat{b})^{n^i p^j} = (\hat{x}^{n^i}+\hat{b})^{p^j} = \hat{x}^{n^i p^j} + \hat{b}$$

em $\mathbb{Z}_p[x]/(x^r-1)$, o que completa a demonstração do lema.

Estamos prontos para provar o resultado que está na raiz do algoritmo AKS. Trata-se de uma variante, devida a H. W. Lenstra Jr, do teorema originalmente obtido por Agrawal, Kayal e Saxena. Este teorema deve ser comparado ao corolário 4.4 da página 64. Lembre-se que o obstáculo a construir um teste de primalidade rápido a partir daquele corolário estava no alto custo de calcular $(x+\overline{b})^n$ em $\mathbb{Z}_n[x]$. A idéia por trás do próximo teorema é que este cálculo pode ser substituído pelo de $(\widehat{x}+\widehat{b})^n$ em $\mathbb{Z}_n[x]/(x^r-1)$, para uma quantidade suficientemente grande de primos r. Note que este último cálculo tem um custo menor que o anterior, visto que as potências de x estão limitadas a x^{r-1} pela congruência.

TEOREMA 5.2. *Sejam $n \geq 2$ e $r \geq 1$ inteiros positivos, e seja $S = \{1, 2, \ldots, r\}$. Suponha que:*

(i) r é um primo que não divide n;
(ii) $\operatorname{mdc}(n, b-b') = 1$, quaisquer que sejam b e b' distintos em S;
(iii) a desigualdade

$$\binom{2r-2}{r} \geq n^{2d\lfloor\sqrt{(r-1)/d}\rfloor}$$

vale para qualquer inteiro positivo d que divide $(r-1)/\nu$, onde ν é a ordem de n em $U(r)$; e
(iv) a igualdade

$$(\hat{x}+\hat{b})^n = \hat{x}^n + \hat{b}$$

é verificada em $\mathbb{Z}_n[x]/(x^r-1)$ para cada $b \in S$.

Então n é uma potência de primo.

DEMONSTRAÇÃO. Seja $p \geq 2$ um primo que divide n. Como n e r são primos entre si, e p divide n, então $\mathrm{mdc}(r,p) = 1$. Portanto, as classes de $\overline{n}$ e $\overline{p}$ são inversíveis em $\mathbb{Z}_r$. Podemos, então, construir o grupo quociente $U(r)/\langle \overline{n}, \overline{p} \rangle$. Digamos que este grupo tem ordem d. Pelo teorema de Lagrange, a ordem ν de $\overline{n}$ em $U(r)$ divide a ordem de $\langle \overline{n}, \overline{p} \rangle$, que é igual a $(r-1)/d$. Portanto, d divide $(r-1)/\nu$ em conformidade com o que estamos assumindo na hipótese (iii).

Suponhamos que $m_1, \ldots, m_d$ são inteiros que representam as d classes que correspondem aos elementos distintos do grupo quociente $U(r)/\langle \overline{n}, \overline{p} \rangle$. Seja h um fator irredutível do polinômio $x^r - 1$ em $\mathbb{Z}_p[x]$. Pelo corolário 3.8, da página 57, o anel quociente $K = \mathbb{Z}_p[x]/(h)$ é um corpo finito. Denotaremos por G o subgrupo do grupo multiplicativo $K^\star$ gerado pelos elementos $\widehat{x}^{m_i} + \widehat{b}$, onde $1 \leq i \leq d$ e $b \in S$.

A maior parte da demonstração consiste em mostrar que o grupo G é "grande o suficiente". Mais precisamente:

$$\#(G) \geq n^{2\lfloor \sqrt{(r-1)/d} \rfloor}, \tag{26}$$

Por isso vamos completar a demonstração do teorema supondo que esta cota inferior para a ordem de G é verdadeira. Só depois provaremos a cota detalhadamente.

DEMONSTRAÇÃO DO TEOREMA 5.2 USANDO A COTA (26). Considere os produtos

$$n^i p^j \ \text{ para } \ 0 \leq i, j \leq \lfloor \sqrt{(r-1)/d} \rfloor.$$

Em particular,

$$1 \leq n^i p^j \leq n^{2\lfloor \sqrt{(r-1)}/d \rfloor}, \tag{27}$$

já que $p \leq n$. Cada classe $\overline{n^i p^j} \in \mathbb{Z}_r$ pertence ao subgrupo $\langle \overline{n}, \overline{p} \rangle$ de $U(r)$. Mas,

$$d = \frac{\#U(r)}{\#\langle \overline{n}, \overline{p} \rangle} = \frac{(r-1)}{\#\langle \overline{n}, \overline{p} \rangle},$$

de modo que o subgrupo $\langle \overline{n}, \overline{p} \rangle$ tem ordem $(r-1)/d$. Portanto, não pode haver mais do que $(r-1)/d$ elementos distintos da forma $\overline{n^i p^j}$ em $\mathbb{Z}_r$.

Por outro lado, como i e j variam a partir de 0 (e não de 1), há um total de $(\lfloor \sqrt{(r-1)/d} \rfloor + 1)^2$ pares

$$(i,j) \ \text{ com } \ 0 \leq i, j \leq \lfloor \sqrt{(r-1)/d} \rfloor.$$

Isto significa que as classes $\overline{n^i p^j}$ não podem ser todas distintas em $\mathbb{Z}_r$. Portanto, existem pares

$$(k_1, \ell_1) \neq (k_2, \ell_2) \ \text{ tais que } \ n^{k_1} p^{\ell_1} \equiv n^{k_2} p^{\ell_2} \pmod{r}. \tag{28}$$

Digamos que $u = n^{k_1} p^{\ell_1}$ e que $v = n^{k_2} p^{\ell_2}$. Suponhamos que $v \geq u$. Como $u \equiv v \pmod{r}$, então $v = u + qr$, para algum inteiro positivo q. Mas,

$$\hat{x}^{u+qr} = \hat{x}^u (\hat{x}^r)^q = \hat{x}^u \ \text{ em } \ \mathbb{Z}_p[x]/(x^r - 1),$$

de modo que

$$\hat{x}^v = \hat{x}^u \text{ em } \mathbb{Z}_p[x]/(x^r - 1).$$

Como $h(x)$ divide $x^r - 1$, concluímos que

$$\hat{x}^v = \hat{x}^u \text{ em } K = \mathbb{Z}_p[x]/(h). \tag{29}$$

Porém, por (iv) e pelo lema 5.1,

$$\prod_{b\in S}(\widehat{x}^{m_i} + \widehat{b})^u = \prod_{b\in S}(\widehat{x}^{m_i} + \widehat{b})^{n^{k_1}p^{\ell_1}} = \prod_{b\in S}(\widehat{x}^u)^{m_i} + \widehat{b},$$

de modo que a igualdade (29) implica que

$$\prod_{b\in S}(\widehat{x}^{m_i} + \widehat{b})^u = \prod_{b\in S}(\widehat{x}^{m_i} + \widehat{b})^v.$$

Como estes produtos geram G, temos que $g^v = g^u$ em K para todo $g \in G$. Em particular, todos os elementos de $G \cup \{0\}$ são soluções da equação polinomial $y^v - y^u$ em K.

Contudo, a desigualdade (27) implica que

$$(v - u) + 1 \leq n^{2\lfloor\sqrt{(r-1)}/d\rfloor};$$

e combinando esta desigualdade com (26), obtemos

$$\#(G) \geq (v - u) + 1.$$

Como $G \subset K$, concluímos que o polinômio

$$y^v - y^u = y^u(y^{v-u} - 1)$$

tem, pelo menos, $(v - u) + 1$ soluções não nulas em K. Mas, isto não pode acontecer se $v - u > 0$, pelo corolário 3.7 da página 57. Logo

$$n^{k_1}p^{\ell_1} = u = v = n^{k_2}p^{\ell_2}.$$

Porém, $k_1 = k_2$ implica que $p^{\ell_1} = p^{\ell_2}$; donde $(k_1, \ell_1) = (k_2, \ell_2)$, o que contradiz a escolha destes pares. Logo $k_1 \neq k_2$. Só que, neste caso,

$$n^{k_1-k_2} = p^{\ell_2-\ell_1},$$

de modo que n tem que ser uma potência de p como queríamos provar.

DEMONSTRAÇÃO DA COTA (26). Para completar todos os detalhes da demonstração do teorema, falta apenas provar a cota dada na equação (26). Faremos isto construindo, explicitamente, uma quantidade suficientemente grande de elementos de G. Com este fim, introduzimos a aplicação que a cada função $e : S \to \mathbb{N}$ associa a d-upla

$$(\widetilde{e}(\widehat{x}^{m_1}), \ldots, \widetilde{e}(\widehat{x}^{m_d})),$$

onde $\widetilde{e}$ é o polinômio definido por

$$\widetilde{e}(y) = \prod_{b\in S}(y + b)^{e(b)}.$$

Como sempre $\widehat{x}$ denota a classe de x em $K = \mathbb{Z}_p[x]/(h)$. Portanto, as entradas desta d-upla são elementos de G.

Mostraremos primeiro que esta aplicação associa d-uplas distintas a certas funções distintas. Sejam e_1 e e_2 funções de S em $\mathbb{N}$. Se as d-uplas associadas a e_1 e e_2 coincidirem, então

$$\widetilde{e_1}(\widehat{x}^{m_i}) = \widetilde{e_2}(\widehat{x}^{m_i}) \text{ em } K, \tag{30}$$

para todo $1 \leq i \leq d$. Porém, por (iv), e pelo lema 5.1,

$$\widetilde{e_1}(\widehat{x}^{m_i})^{n^k p^\ell} = \prod_{b \in S}((\widehat{x}^{m_i} + \widehat{b})^{e_1(b)})^{n^k p^\ell} = \prod_{b \in S}(\widehat{x}^{m_i n^k p^\ell} + \widehat{b})^{e_1(b)} = e_1(\widehat{x}^{m_i n^k p^\ell}),$$

em $\mathbb{Z}_p[x]/(x^r - 1)$, para todo $k, \ell \geq 0$. Entretanto, como h é um fator de $x^r - 1$, qualquer igualdade que vale em $\mathbb{Z}_p[x]/(x^r - 1)$ também vale em $K = \mathbb{Z}_p[x]/(h)$. Logo,

$$\widetilde{e_1}(\widehat{x}^{m_i n^k p^\ell}) = \widetilde{e_2}(\widehat{x}^{m_i n^k p^\ell}) \text{ em } K \text{ para todo } 1 \leq i \leq d. \tag{31}$$

Ponha $g(x) = \widetilde{e_1}(x) - \widetilde{e_2}(x)$. Como $m_1, \ldots, m_d$ são representantes distintos de cada uma das classes de $U(r)/\langle \overline{n}, \overline{p} \rangle$ então, dado um inteiro $1 \leq \alpha \leq r-1$, existem $k, \ell \in \mathbb{N}$, tais que

$$\alpha \equiv m_i n^k p^\ell \pmod{r}.$$

Assim, $\alpha = m_i n^k p^\ell + tr$ para algum inteiro t. Portanto,

$$g(\widehat{x}^\alpha) = g(\widehat{x}^{m_i n^k p^\ell}(\widehat{x}^r)^t) = g(\widehat{x}^{m_i n^k p^\ell}) \text{ em } \mathbb{Z}_p[x]/(x^r - 1),$$

já que $\hat{x}^r = \hat{1}$ em $\mathbb{Z}_p[x]/(x^r - 1)$. Como h é fator de $x^r - 1$, então

$$g(\widehat{x}^\alpha) = g(\widehat{x}^{m_i n^k p^\ell}) \text{ em } \mathbb{Z}_p[x]/(h),$$

donde segue, por (31), que

$$g(\widehat{x}^\alpha) = 0 \text{ em } \mathbb{Z}_p[x]/(h) = K. \tag{32}$$

Considere, agora, o conjunto

$$R = \{\widehat{x}^\alpha : 1 \leq \alpha \leq r-1\} \subseteq K.$$

Como $h(x)$ divide $x^r - 1$, então $\widehat{x}^r = 1$ em K. Contudo, r é primo, de modo que, pelo lema chave da página 33, $\widehat{x}$ tem ordem r em $K^\star$. Em particular, os elementos de R são todos distintos. Mas, por (32), cada elemento de R é raiz de $g(y)$ em K. Então, pelo corolário 3.7 da página 57, temos que $g(y)$ tem grau pelo menos $r-1$ como polinômio de $\mathbb{Z}_p[y]$. Assim, supondo que

$$\sum_{b \in S} e_1(b) \text{ e } \sum_{b \in S} e_2(b) \text{ são menores que } r-1, \tag{33}$$

concluímos que $\widetilde{e_1}(y) = \widetilde{e_2}(y)$ em $\mathbb{Z}_p[y]$. Entretanto, pela hipótese (ii),

$$(y + \overline{b}) - (y + \overline{b'}) = \overline{b} - \overline{b'} \neq 0 \text{ em } \mathbb{Z}_p[y].$$

Logo, dois elementos distintos b e b' de S dão lugar a polinômios irredutíveis distintos $y + \overline{b}$ e $y + \overline{b'}$ em $\mathbb{Z}_p[y]$. Portanto, pelo Teorema da Fatoração Única para polinômios, $\widetilde{e_1}(y) = \widetilde{e_2}(y)$ só pode acontecer, sob a hipótese (33), se as funções e_1 e e_2 forem iguais. Em particular, o conjunto das d-uplas de

elementos de G tem, pelo menos, tantos elementos quantas são as funções $e : S \to \mathbb{N}$ para as quais $\sum_{b \in S} e(b) < r - 1$. Contudo, esta equação linear tem

$$\binom{2r-2}{r}$$

soluções inteiras não negativas, de modo que

$$\#(G)^d \geq \binom{2r-2}{r}.$$

Combinando esta desigualdade com a hipótese (iii), concluímos que

$$\#(G)^d \geq n^{2d\lfloor\sqrt{(r-1)/d}\rfloor},$$

donde a equação (26) segue imediatamente.

2. O algoritmo AKS

Nesta seção enunciamos o algoritmo AKS, conforme a modificação sugerida por Lenstra, e provamos que funciona. Deixaremos o estudo do custo deste algoritmo para a próxima seção. Lembre-se que, se a é um número real, então $\lceil a \rceil$ denota o menor inteiro maior que a.

Algoritmo AKS

Entrada: um inteiro positivo ímpar n.

Saída: *primo* ou *composto*.

Etapa 1: Verifique se n é um quadrado, cubo ou outra potência de um número inteiro. Se for imprima *composto* e páre.

Etapa 2: Calcule

$$N = 2n(n-1)(n^2-1)(n^3-1)\cdots(n^{4\lceil\log_2 n\rceil^2}-1)$$

e determine o menor primo r que não divide N.

Etapa 3: Verifique se n é divisível por algum primo $q < r$. Se for, e se

- $q = n$, imprima *primo* e páre.
- $q < n$, imprima *composto* e páre.

Etapa 4: Verifique se

$$(\hat{x} + \hat{b})^n = \hat{x}^n + \hat{b} \text{ em } \mathbb{Z}_n[x]/(x^r - 1),$$

para cada $b \in S = \{1, 2, \ldots, r\}$. Se a congruência não for verificada para algum $b \in S$, imprima *composto* e páre.

Etapa 5: Se a congruência for verificada para todo $b \in S$, imprima *primo* e páre.

Antes de continuar precisamos verificar se de fato sabemos executar cada uma destas etapas. Mesmo uma leitura superficial mostra que as etapas 2, 4 e

5 não apresentam nenhum problema. Para a etapa 3 podemos usar o crivo de Eratóstenes: geramos os primos de 2 a r e verificamos se dividem ou não n.

Com isto resta-nos apenas considerar a primeira etapa, que aliás nada tem a ver com a teoria que desenvolvemos nos capítulos anteriores. Uma estratégia simples para executá-la consiste em mostrar que a equação $x^k - n = 0$ não tem raiz inteira para nenhum inteiro $k > 0$. Isto pode ser feito, por exemplo, usando-se uma variação do método de Newton, como é explicado no apêndice. Na próxima seção veremos como estimar o maior valor que k pode assumir.

Naturalmente a demonstração do algoritmo será feita a partir do teorema 5.2. É fácil relacionar cada uma das etapas do algoritmo às hipóteses e à conclusão do teorema 5.2, com exceção da segunda. Mas, como veremos a seguir, o objetivo da segunda etapa é forçar n a ter uma ordem alta em $U(r)$; o que por sua vez garante a validade da hipótese (iii) do teorema 5.2.

DEMONSTRAÇÃO DO ALGORITMO AKS. Suponhamos, primeiramente, que n seja primo. Se por acaso n for menor que o primo r encontrado na etapa 2, então o algoritmo o detectará corretamente. Caso contrário, o algoritmo executará a quarta etapa. Como estamos supondo que n é primo então, pelo teorema 4.1 da página 61, a congruência será satisfeita para todos os $b \in S$. Portanto, o algoritmo retornará primo também neste caso.

Reciprocamente, suponhamos que o algoritmo tem saída *primo*. Isto significa que o algoritmo parou na etapa 3, ou na etapa 5. Se parou na terceira etapa afirmando que n é primo, então podemos estar certos de que isto é verdade, porque tudo o que fizemos foi uma busca sistemática pelos primos menores que r. Resta-nos mostrar que, se o algoritmo chegar a executar a etapa 5, então n também é primo.

Mas, para que a etapa 5 seja executada, é preciso que o algoritmo determine, na etapa 2, um primo r que não divide

$$N = 2n(n-1)(n^2-1)(n^3-1)\cdots(n^{4\lceil \log_2 n\rceil^2}-1).$$

Em particular, isto garante que a hipótese (i) do teorema 5.2 é satisfeita. Por outro lado, pela etapa 3 do algoritmo, n não é divisível por nenhum primo menor que r. Como $S = \{1, 2, \ldots, r\}$, isto significa que a hipótese (ii) do teorema 5.2 também é satisfeita. Levando em conta que a hipótese (iv) do teorema 5.2 é verificada diretamente nas etapas 4 e 5, falta-nos apenas considerar hipótese (iii).

Mas, r não divide N, de modo que

$$n^k \not\equiv 1 \pmod{r} \text{ para todo } 1 \le k \le 4\lceil \log_2 n\rceil^2.$$

Portanto, a ordem ν de $\overline{n}$ em $U(r)$ tem que ser maior do que $4\lceil \log_2 n\rceil^2$. Seja, agora, d um inteiro positivo que divide $(r-1)/\nu$. Então,

$$d \le \frac{(r-1)}{\nu} < \frac{(r-1)}{4\lceil \log_2 n\rceil^2} \le \frac{(r-1)}{4(\log_2 n)^2}. \tag{34}$$

Mas,

$$2d\lfloor\sqrt{\frac{(r-1)}{d}}\rfloor \leq 2d\sqrt{\frac{(r-1)}{d}} = \sqrt{4d(r-1)}.$$

Combinando esta última desigualdade com (34), obtemos

$$2d\lfloor\sqrt{\frac{(r-1)}{d}}\rfloor \leq \frac{(r-1)}{\log_2 n},$$

de modo que

$$n^{2d\lfloor\sqrt{(r-1)/d}\rfloor} \leq n^{(r-1)/\log_2 n} = 2^{(r-1)},$$

pois $\log_2(n)\log_n(2) = 1$. Por outro lado,

$$\binom{2(r-1)}{r} \geq 2^{r-1},$$

já que $r \geq 2$; veja exercício 3. Portanto,

$$\binom{2r-2}{r} \geq 2^{r-1} \geq n^{2d\lfloor\sqrt{(r-1)/d}\rfloor}.$$

Mas esta é a hipótese (iii) do teorema 5.2.

Provamos, até aqui, que se o algoritmo retorna *primo* então todas as hipóteses do teorema 5.2 são satisfeitas. Portanto, pelo teorema, $n = p^e$, para algum primo $p > 1$. Mas n não pode ser igual a nenhuma potência de inteiro com expoente igual ou maior do que 2, pela etapa 1. Logo $e = 1$ e n é primo.

3. Custo do algoritmo

Estamos prontos para proceder à análise do custo do algoritmo AKS. Faremos isto considerando cada etapa do algoritmo separadamente e mostrando que pode ser executada em tempo polinomial. Naturalmente, isto se restringe às quatro primeiras etapas, uma vez que não estamos executando nenhuma operação aritmética na etapa 5.

Análise da etapa 1. Na etapa 1 do algoritmo determinamos se existem inteiros positivos $a > 1$ e $k > 1$ de modo que $a^k = n$. A primeira coisa a fazer é descobrir qual é o maior valor que k pode assumir. Mas k será máximo quando a for mínimo; ou seja, se $a = 2$. Neste caso teríamos $n = 2^k$, donde $k = \log_2 n$. Logo podemos concluir que $2 \leq k \leq \log_2 n$.

Um método pelo qual podemos determinar se n é ou não a k-ésima potência de um inteiro está descrito detalhadamente no apêndice. A análise deste método mostra que seu custo é inferior a $O((\log_2 n)^6)$. Como o algoritmo precisa ser executado $\log_2 n$ vezes, obtemos um custo total inferior a $O((\log_2 n)^7)$ para esta etapa.

Análise da etapa 2. Para provar que a etapa 2 pode ser executada em tempo polinomial, basta mostrar que o menor valor possível do primo r é limitado superiormente por um polinômio em $\log_2 n$.

Seja $m = 4\lceil \log_2 n \rceil^2$. Tomando logaritmos na base 2, temos que

$$\log_2 N = 1 + \log_2 n + \sum_{i=1}^{m} \log_2(n^i - 1).$$

Mas $\log_2(n^i - 1) \leq i \log_2 n$, donde

$$\log_2 N \leq 1 + \log_2 n + \sum_{i=1}^{m} i \log_2 n \leq 1 + \left(1 + \frac{m(m-1)}{2}\right) \log_2 n. \tag{35}$$

Como $m < 4(\log_2 n)^2 + 1$, concluímos que existe um inteiro

$$\log_2 N < k \leq 2 + \left(1 + \frac{(4(\log_2 n)^2 + 1)4(\log_2 n)^2}{2}\right) \log_2 n. \tag{36}$$

Note que somamos 1 ao lado direito da desigualdade (35) para poder garantir que existe um inteiro k no intervalo desejado. Em particular, $N < 2^k$. Mas, pelo teorema 1.2 o produto dos primos menores ou iguais a $2k^2$ é maior ou igual do que 2^k. Logo, tem que existir pelo menos um primo $r \leq 2k^2$ que não divide N. Observe que esta última desigualdade e (35) implicam que r é $O((\log_2 n)^{10})$.

Portanto, podemos achar r simplesmente testando os inteiros, a partir de 2, para descobrir qual é o primeiro que não divide N. Para isso, efetuamos uma divisão para cada inteiro. Como todos estes inteiros serão menores que $2k^2$ cada divisão tomará um tempo $O((\log_2 \log_2 n)^2)$. Como $r \leq 2k^2$, não precisamos efetuar mais do que $2k^2$ divisões, o que dá um custo total de $O(k^2(\log_2 \log_2 n)^2)$, que é inferior a $O((\log_2 n)^{12})$, pela desigualdade (36). Mas isto significa que a etapa 2 também pode ser executada a um custo polinomial.

Observe que podemos facilmente melhorar nossa avaliação do tempo necessário para achar r se usarmos uma cota melhor no teorema de Chebyshev; veja, por exemplo, **[Hardy e Wright, capítulo XXII]**.

Análise da etapa 3. Na execução desta etapa usamos apenas o crivo de Eratóstenes para listar os primos $q < r$. Cada vez que um desses primos for descoberto pelo crivo, testamos para ver se divide n. Portanto, o custo desta etapa é igual ao custo de executar o crivo até r, somado ao custo total das divisões.

De acordo com a análise feita na seção 3 do capítulo 1, o crivo de Eratóstenes tem custo

$$O(r^2) = O((\log_2 n)^{20}),$$

se levamos em conta a ordem de r, obtida na análise da etapa 2. Já o custo incorrido para dividir n por um inteiro menor que r é $O((\log_2 n)^2)$. Como não precisamos efetuar mais do que r divisões, o custo total desta etapa fica abaixo de $O((\log_2 n)^{20})$. Portanto, mais uma vez, o tempo de execução é polinomial.

Análise da etapa 4. Esta etapa requer bastante cuidado. Nosso objetivo é calcular uma n-ésima potência no anel $\mathbb{Z}_n[x]/(x^r - 1)$. Usando o algoritmo

de exponenciação descrito na seção 4 do capítulo 5, sabemos que isto pode ser feito ao custo de, aproximadamente, $\log_2 n$ quadrados. Mas, quanto custa calcular um quadrado em $\mathbb{Z}_n[x]/(x^r - 1)$?

Digamos que desejamos calcular $\widehat{f}^2$ onde $\widehat{f} \in \mathbb{Z}_n[x]/(x^r - 1)$. Sabemos, da definição de anel quociente, que $f \in \mathbb{Z}_n[x]$ pode ser escolhido de modo que o seu grau seja menor que r e seus coeficientes sejam inteiros não negativos menores que n. Pela proposição 3.1 da página 45, o custo de calcular f^2 em $\mathbb{Z}_n[x]$ é $O(r^2(\log_2 n)^2)$.

Entretanto, o polinômio f^2 pode ter grau maior do que r, de modo que precisamos ainda reduzi-lo módulo $x^r - 1$ para achar $\widehat{f}^2$. Para isso, precisamos reduzir cada x^j para o qual $j > r$. Fazemos isto dividindo j por r. Digamos que o quociente seja β e o resto seja ρ. Então,

$$\hat{x}^j = (\hat{x}^r)^\beta \hat{x}^\rho = \hat{x}^\rho \text{ em } \mathbb{Z}_n[x]/(x^r - 1).$$

O custo para realizar isto é o de efetuar a divisão de dois números menores que $2(r - 1)$. Como há, no máximo, r termos de grau maior ou igual a r em f^2, o custo de realizar todas estas divisões é da ordem de $O((\log_2 r)^3) = O((\log_2 \log_2 n)^3)$.

Completadas estas etapas, verificamos que cada termo de f^2 que tenha grau j, com $r \leq j \leq 2(r - 1)$, produz exatamente um termo de grau menor que r. Resta-nos somar estes novos termos com os termos de grau menor que r que já existiam em f^2, mesmo antes da redução. Temos, assim, mais r somas a efetuar, a um custo total de $O(r(\log_2 n)^2)$.

Como $r^2(\log_2 n)^2$ tem ordem maior do que $(\log_2 \log_2 n)^3$ e também do que $r(\log_2 n)^2$, concluímos que calcular $\widehat{f}^2$ tem custo

$$O(r^2(\log_2 n)^2) = O((\log_2 n)^{22}),$$

já que, pela discussão da etapa 2, r tem ordem $(\log_2 n)^{10}$. Mas o que queremos mesmo é calcular r potências n-ésimas, uma para cada elemento de S. Portanto, o custo total desta etapa é de

$$O(r(\log_2 n)r^2(\log_2 n)^2) = O((\log_2 n)^{33}),$$

Mais uma vez, o custo é polinomial.

Isto conclui a análise do custo do algoritmo, e comprova que se trata de um algoritmo polinomial. Observe que, de todas as etapas do algoritmo AKS, aquela com o custo mais alto é a quarta. Isto significa que, se desejamos tornar o algoritmo mais eficiente, precisamos de um procedimento melhor para calcular potências em $\mathbb{Z}_n[x]/(x^r - 1)$. Este problema é discutido em mais detalhes no epílogo. Tendo dito isto, precisamos admitir que nossa análise do custo do AKS sofreu–e muito!–com a limitação imposta pela decisão de manter o argumento tão simples quanto possível. Uma análise mais cuidadosa do custo das várias etapas do algoritmo leva aos valores relacionados na tabela da próxima página. Levando em conta os melhores valores, o algoritmo tem custo inferior a $O((\log_2 n)^{13})$.

Etapa	Custo Otimizado	Custo Acima
1	$O((\log_2 n)^3)$	$O((\log_2 n)^7)$
2	$O(\log_2 n)^5$	$O((\log_2 n)^{12})$
3	$O((\log_2 n)^{10})$	$O((\log_2 n)^{20})$
4	$O((\log_2 n)^{12} \log_2(\log_2 n) \log_2 \log_2 \log_2 n)$	$O((\log_2 n)^{33})$

TABELA 1. Custo das etapas do AKS

4. Exercícios

1. Seja m um inteiro positivo. Mostre que a equação $x_1 + \cdots + x_s = m$ tem
   (a) $\binom{m-1}{s-1}$ soluções inteiras positivas, e
   (b) $\binom{m+s-1}{s-1}$ soluções inteiras não negativas.

   SUGESTÃO: Reduza (b) a (a) provando que $y_1 + \cdots + y_s = m + s$ tem tantas soluções positivas quantas são as soluções não negativas da equação dada.

2. Mostre que a equação $x_1 + \cdots + x_s < m$ tem $\binom{m+s-1}{s}$ soluções inteiras não negativas.

   SUGESTÃO: Reduza ao exercício anterior, provando que $x_1 + \cdots + x_s + z = m - 1$ tem tantas soluções não negativas quantas são as soluções não negativas da equação dada.

3. Mostre que
$$\binom{2(r-1)}{r} \geq 2^{r-1}$$
   para todo $r \geq 3$.

   SUGESTÃO:
$$\binom{2(r-1)}{r} = \frac{2r-2}{r-2} \cdot \frac{2r-3}{r-3} \cdots \frac{r+1}{1},$$
   é um produto de r termos. Por outro lado,
$$\frac{2r-j}{r-j} = 2 + \frac{j}{r-j} \geq 3,$$
   se $j \geq \lfloor r/2 \rfloor$. Isto basta para provar a desigualdade se $r \geq 4$ (por quê?). Um cálculo direto mostra que a desigualdade também vale se $r = 3$.

4. Sejam $n \geq 2$ e $r \geq 1$ inteiros positivos, e seja $S = \{1, 2, \ldots, r\}$. Suponha que
   (i) r é um primo que não divide n;
   (ii) $\overline{n}$ gera $U(r)$;
   (iii) $\mathrm{mdc}(n, b - b') = 1$, quaisquer que sejam b e b' distintos em S;

(iv) vale a desigualdade
$$\binom{2r-2}{r} \geq n^{2d\lfloor\sqrt{(r-1)}\rfloor};$$
(v) e a igualdade
$$(\hat{x}+\hat{b})^n = \hat{x}^n + \hat{b}$$
é verificada em $\mathbb{Z}_n[x]/(x^r-1)$ para cada $b \in S$.

Prove que n é uma potência de primo. De que maneira as hipóteses acima diferem das hipóteses do teorema 5.2?

5. Modifique a demonstração do teorema 5.2 de maneira que se aplique ao caso em que a hipótese (i) é substituída por $\mathrm{mdc}(n,r)=1$ e $r-1$ é substituído por $\phi(r)$ na hipótese (iii).

6. Segundo um teorema provado por M. Goldfeld em 1969, há muitos primos r para os quais $r-1$ tem um divisor primo maior que $r^{2/3}$. Como conseqüência disto pode-se mostrar que existem q e r suficientemente pequenos tais que
$$q \geq 4\lceil\sqrt{r}\rceil\lceil\log_2 n\rceil \text{ e } n^{(r-1)/q} \not\equiv 0,1 \pmod{r}.$$
Prove que, sob estas hipóteses, o conjunto $S = \{1,2,\ldots,2\lceil\sqrt{r}\rceil\lceil\log_2 n\rceil\}$ satisfaz a desigualdade
$$\binom{s+q-1}{s} \geq n^{2\lfloor\sqrt{r}\rfloor},$$
onde s é o número de elementos de S. Esta condição substituía a hipótese (iv) do teorema 5.2 no enunciado original de Agrawal, Kayal e Saxena, conforme a interpretação de D. Bernstein.

7. Determine o custo das etapas 3 e 4 do algoritmo, sabendo-se que a etapa 2 pode ser executada em tempo $O((\log_2 n)^5)$.

8. Neste exercício discutimos uma maneira de acelerar o cálculo da etapa 4 do algoritmo AKS. Como vimos na seção 3, o problema se reduz a calcular o quadrado de um polinômio em $\mathbb{Z}_n[x]/(x^r-1)$. Seja f um polinômio em $\mathbb{Z}[x]$ que representa a classe em $\mathbb{Z}_n[x]/(x^r-1)$ da qual queremos calcular o quadrado. Podemos supor que os coeficientes de f são inteiros entre 0 e $n-1$. Seja $k = \lceil\log_2(rn^2)\rceil$. Denote por g o polinômio de menor grau em $\mathbb{Z}[x]$ que representa a classe de f^2 em $\mathbb{Z}[x]/(x^r-1)$.
(a) Mostre que os coeficientes de g são todos menores que $rn^2 \leq 2^k$.
(b) Considere a aplicação
$$\psi : \mathbb{Z}[x]/(x^r-1) \to \mathbb{Z}_{2^{kr}-1},$$
definida por $\psi(\hat{h}(x)) = h(2^k)$, onde $h \in \mathbb{Z}[x]$. Mostre que
$$\psi(\widehat{h_1 h_2}) = \psi(\widehat{h_1})\psi(\widehat{h_2}),$$
onde $\widehat{h_1}, \widehat{h_2} \in \mathbb{Z}[x]/(x^r-1)$.

(c) Mostre que ψ é injetiva.

Portanto, calculamos $\widehat{h_1 h_2}$ efetuando o produto $\psi(\widehat{h_1})\psi(\widehat{h_2})$ em $\mathbb{Z}_{2^{kr}-1}$ e aplicando ψ^{-1} ao resultado. O polinômio obtido desta maneira pode, então, ser reduzido módulo n.

9. Determine o custo de execução do algoritmo descrito no exercício anterior.

Epílogo

Como foi dito no prefácio, e várias vezes repetido ao longo do texto, este livro não foi escrito com o objetivo de dar uma exposição completa de tudo o que se sabe sobre o algoritmo AKS. Muito menos de tratar de maneira sistemática o problema de determinar se um dado número é primo ou composto. Na verdade, é tal a atividade em torno das idéias introduzidas no AKS, que qualquer tentativa de cobrir sistematicamente o que já foi feito estaria obsoleta antes mesmo de ser impressa.

Por outro lado, além de manter o argumento o mais elementar possível, o livro aborda o algoritmo AKS de um ponto de vista meramente teórico. Isto significa que, além de pouco termos falado a respeito da implementação do algoritmo, nossa análise do próprio AKS deixa muito a desejar. Por exemplo, discutimos somente uma das várias versões do algoritmo, e nada dissemos sobre os vários aperfeiçoamentos conjecturais que foram propostos, alguns dos quais reduzem significativamente o custo total do algoritmo. Para falar a verdade, nossa análise de custo foi tão grosseira que não faria muito sentido embarcar nestes refinamentos.

Seja como for, não podemos simplesmente encerrar o livro sem ao menos apontar algumas referências bibliográficas onde estas questões são discutidas com detalhe, e é esta a missão do epílogo.

Para começar, nossa discussão dos testes de primalidade anteriores ao AKS foi extramamente superficial. Por exemplo, nada dissemos de substancial sobre maneiras pelas quais se podia provar a primalidade de um número grande. Como mencionado na introdução, alguns dos primeiros métodos para certificar que um número é primo foram introduzidos por E. Lucas, e depois aperfeiçoados por D. H. Lehmer. Nas décadas finais do século XX estes testes foram generalizados a perder de vista, dando origem a métodos bastante eficientes para determinar a primalidade de um número. Mais detalhes sobre os testes de primalidade disponíveis antes de 2002 podem ser encontrados em **[Crandall e Pomerance 2002]** e **[Ribenboim 2001]**. Este último livro é uma mina de toda espécie de informação curiosa sobre números primos.

Ainda hoje, a maneira mais eficaz de verificar se um número de 100, ou 200 algarismos, é primo é utilizar o Teste da Curva Elíptica, descoberto na década de 1990. Uma *curva elíptica* é uma curva plana (não singular) descrita por uma equação de grau 3. Sabe-se, desde o século XIX, que é possível definir

uma adição no conjunto de pontos desta curva. Por sua vez, esta operação determina uma estrutura de grupo abeliano sobre a curva. O Teste da Curva Elíptica adapta a este grupo abeliano a estratégia já utilizada por Lucas no grupo $U(n)$.

O estudo das curvas elípticas, que combina técnicas de álgebra, análise, teoria de números e geometria algébrica, é uma das áreas mais ricas da matemática. Um de seus sucessos mais recentes foi a demonstração por A. Wiles do Último Teorema de Fermat, já mencionada na introdução. Na verdade, o que Wiles provou foi uma conjectura sobre curvas elípticas, formulada na década de 1950, e que, segundo já se sabia, tinha como conseqüência o resultado enunciado por Fermat. Entretanto, tal é a sua abrangência, que as curvas elípticas também têm aplicações práticas que incluem, além do teste de primalidade já mencionado, um algoritmo de fatoração para números inteiros e vários algoritmos criptográficos.

Uma exposição dos aspectos mais elementares da teoria das curvas elípticas pode ser encontrada no excelente **[Silverman e Tate 1992]**, que inclui uma discussão do algoritmo de fatoração. O Teste da Curva Elíptica é analisado no Capítulo 7 de **[Crandall e Pomerance 2002]**. A utilização de curvas elípticas em criptografia é amplamente discutida em **[Blake, Seroussi e Smart 2000]**. Para uma comparação entre o AKS e o Teste da Curva Elíptica consulte **[Morain 2002]**.

Passando ao AKS, uma exposição mais sofisticada da mesma versão apresentada neste livro é dada em **[Schoof 2003]**. Várias versões diferentes do AKS, incluindo a original, são descritas no artigo **[Bernstein 2003a]**, que foi a principal fonte deste livro. Nele você também vai achar uma discussão detalhada da eficiência do algoritmo AKS e de sua implementação. Um tratamento mais minucioso de uma implementação bastante otimizada pode ser encontrado em **[Crandall e Papadopoulos 2003]**.

A principal falha deste livro, no que diz respeito à implementação do AKS, é que nada dizemos sobre procedimentos eficientes para executar a etapa 4 do algoritmo. Como vimos, a exponenciação de polinômios, executada nesta etapa, é a operação de custo mais alto de todo o algoritmo. Por isso, substituir o procedimento usual de exponenciação por outro mais rápido terá um impacto imediato no tempo de execução do AKS. O mais forte candidato a método eficaz para a exponenciação de polinômios é o algoritmo baseado na Transformada Rápida de Fourier, geralmente conhecida pela sigla inglesa FFT. Este algoritmo é descrito em detalhes em **[von zur Gathen e Gerhard 1999, Capítulo 8]**. Mas esta não é a única possibilidade. Por exemplo, a implementação de Crandall e Papadopoulos mencionada acima utiliza outro algoritmo, conhecido como *multiplicação por segmentação binária*, cuja descrição pode ser encontrada em **[Crandall e Pomerance 2002, p. 467]**. Veja também o exercício 8 do capítulo 5.

Além das várias versões do AKS, que começaram a surgir tão logo o artigo de Agrawal, Kayal e Saxena foi postado na web, devemos mencionar uma

gama de novos algoritmos por ele inspirados. De um lado, temos refinamentos do AKS que são mais eficientes para certas classes de primos **[Berrizbeitia 2002]**; de outro, propostas de algoritmos probabilísticos, extremamente eficientes, que utilizam uma estratégia semelhante à do AKS **[Bernstein 2003b]**, **[Mihăilescu e R. M. Avanzi 2003]**.

Em matemática, a solução de um problema importante raramente representa um fim, ou uma conclusão. Freqüentemente o que acontece é que uma nova Caixa de Pandora, cheia de surpresas, foi aberta. Com isso, logo surge um grupo de pessoas que se propõe a explorar as conseqüências deste novo resultado. Este é, certamente, o caso com o AKS. De fato, a simplicidade deste algoritmo traz à tona uma pergunta ainda mais importante, e que pode ser o prenúncio de grandes resultados que estão por vir:

> Que outros algoritmos, igualmente simples, ainda não fomos espertos bastante para descobrir?

Apêndice

Neste apêndice descrevemos um algoritmo que, dado um inteiro positivo, determina se alguma raiz n-ésima de um inteiro dado é inteira. Além de demonstrar que o algoritmo funciona, mostraremos que tem custo polinomial.

1. O método de Newton

Nesta seção estabelecemos uma espécie de método de Newton módulo p, que será a base do algoritmo da próxima seção. Começamos com dois lemas básicos sobre polinômios. Ao longo de todo este apêndice denotaremos por $f'(x)$ a derivada de um polinômio $f(x)$ com coeficientes inteiros.

LEMA A.1. *Seja $g(x)$ um polinômio com coeficientes inteiros, e seja $\alpha \in \mathbb{Z}$. Então, $(x-\alpha)^2$ divide $g(x)$ se, e somente se, $g(\alpha) = g'(\alpha) = 0$.*

DEMONSTRAÇÃO. Suponhamos, em primeiro lugar, que $(x-\alpha)^2$ divide $g(x)$. Neste caso, existe um polinômio $q(x) \in \mathbb{Z}[x]$, tal que

$$g(x) = (x-\alpha)^2 q(x).$$

Derivando $g(x)$, obtemos

$$g'(x) = 2(x-\alpha)q(x) + (x-\alpha)^2 q'(x).$$

Donde é claro que $g(\alpha) = g'(\alpha) = 0$.

Reciprocamente, suponhamos que $g(\alpha) = g'(\alpha) = 0$. Mas, pelo lema 3.3 da página 48, se $g(\alpha) = 0$, então $(x-\alpha)$ divide $g(x)$. Logo, existe $h(x) \in \mathbb{Z}[x]$ tal que $g(x) = (x-\alpha)h(x)$. Derivando esta última expressão para $g(x)$, obtemos

$$g'(x) = h(x) + (x-\alpha)h'(x).$$

Portanto, $h(\alpha) = g'(\alpha) = 0$. Usando novamente o lema 3.3 da página 48, concluímos que existe $q(x) \in \mathbb{Z}[x]$ tal que $h(x) = (x-\alpha)q(x)$. Assim,

$$g(x) = (x-\alpha)h(x) = (x-\alpha)^2 q(x),$$

como queríamos mostrar.

O segundo lema corresponde à fórmula de Taylor para um polinômio.

LEMA A.2. *Seja $f(x) \in \mathbb{Z}[x]$ e α um inteiro qualquer. Então*

$$f(x) = f(\alpha) + f'(\alpha)(x-\alpha) + h(x)(x-\alpha)^2,$$

para algum $h(x) \in \mathbb{Z}[x]$.

DEMONSTRAÇÃO. Pelo lema anterior, basta mostrar que

$$g(x) = f(x) - f(\alpha) - f'(\alpha)(x-\alpha),$$

satisfaz $g(\alpha) = g'(\alpha) = 0$. Mas

$$g(\alpha) = f(\alpha) - f(\alpha) - f'(\alpha)(\alpha-\alpha) = 0,$$

ao passo que, $g'(x) = f'(x) - f'(\alpha)$, donde $g'(\alpha)$ também é zero, completando a demonstração do lema.

O resultado central deste apêndice é a seguinte versão módulo p do método de Newton.

TEOREMA A.3. *Sejam α e $p > 0$ inteiros, onde p é primo. Suponhamos que $f(x) \in \mathbb{Z}[x]$ é tal que $f'(\alpha)$ é inversível módulo p e que $f(\alpha) \equiv 0 \pmod{p^k}$, para algum inteiro $k > 0$. Se*

$$\beta \equiv \alpha - f(\alpha)(f'(\alpha))^{-1} \pmod{p^{2k}}, \tag{37}$$

então

(i) $\beta \equiv \alpha \pmod{p^k}$*;*
(ii) $f(\beta) \equiv 0 \pmod{p^{2k}}$*; e*
(iii) $f'(\beta)$ *é inversível módulo* p*.*

DEMONSTRAÇÃO. Como p é primo, então

$$\mathrm{mdc}(a, p^k) = 1 \text{ se, e somente se, } \mathrm{mdc}(a, p) = 1,$$

qualquer que seja $a \in \mathbb{Z}$. Em particular, $f'(\alpha)$ é inversível módulo p^{2k} e a equação (37) está bem definida.

Para provar (i) observe primeiro que, como

$$\beta \equiv \alpha - f(\alpha)(f'(\alpha))^{-1} \pmod{p^{2k}},$$

então a mesma congruência vai valer módulo p^k. Logo,

$$\beta \equiv \alpha - f(\alpha)(f'(\alpha))^{-1} \equiv \alpha \pmod{p^k},$$

já que $f(\alpha) \equiv 0 \pmod{p^k}$, por hipótese.

Para provar (ii) fazemos $x = \beta$ na fórmula dada pelo lema A.2, obtendo

$$f(\beta) = f(\alpha) + f'(\alpha)(\beta-\alpha) + h(\alpha)(\beta-\alpha)^2,$$

onde $h(x) \in \mathbb{Z}[x]$. Como p^k divide $\beta - \alpha$ por (i), então $(\beta-\alpha)^2 \equiv 0 \pmod{p^{2k}}$, de modo que

$$f(\beta) \equiv f(\alpha) + f'(\alpha)(\beta-\alpha) \pmod{p^{2k}}.$$

Contudo, $\beta - \alpha \equiv -f(\alpha)(f'(\alpha))^{-1} \pmod{p^{2k}}$ por (37), donde

$$f(\beta) \equiv f(\alpha) + f'(\alpha)(\beta-\alpha) \equiv f(\alpha) + f'(\alpha)(-f(\alpha)(f'(\alpha))^{-1}) \equiv 0 \pmod{p^{2k}},$$

o que prova (ii).

Finalmente, (i) implica que $\beta \equiv \alpha \pmod{p}$, de modo que $f'(\beta) \equiv f'(\alpha) \pmod{p}$. Então, (iii) segue do fato de $f'(\alpha)$ ser inversível módulo p por hipótese.

O próximo resultado estabelece a unicidade de um zero de um polinômio módulo uma potência de primo a partir de certas condições iniciais. Observe que as hipóteses são essencialmente as mesmas do teorema anterior.

TEOREMA A.4. *Sejam α e $p > 0$ inteiros, onde p é primo. Suponhamos que $f(x) \in \mathbb{Z}[x]$ é tal que $f'(\alpha)$ é inversível módulo p e que $f(\alpha) \equiv 0 \pmod{p}$. Sejam β_1, β_2 e $k > 0$ inteiros tais que*

(i) $\beta_1 \equiv \beta_2 \equiv \alpha \pmod{p}$, *e*

(ii) $f(\beta_1) \equiv f(\beta_2) \equiv 0 \pmod{p^k}$,

então $\beta_1 \equiv \beta_2 \pmod{p^k}$.

DEMONSTRAÇÃO. Pela fórmula do lema A.2

$$f(\beta_2) - f(\beta_1) = (\beta_2 - \beta_1)(f'(\beta_1) + h(\beta_1)(\beta_2 - \beta_1)), \tag{38}$$

onde $h(x) \in \mathbb{Z}[x]$. Porém,

$$f'(\beta_1) + h(\beta_1)(\beta_2 - \beta_1) \equiv f'(\beta_1) \equiv f'(\alpha) \pmod{p}$$

já que $\beta_1 \equiv \beta_2 \equiv \alpha \pmod{p}$ por (i). Mas, $f'(\alpha)$ é inversível módulo p, de modo que também será inversível módulo p^k. Portanto, $f'(\beta_1) + h(\beta_1)(\beta_2 - \beta_1)$ é inversível módulo p^k. Digamos que seu inverso é u. Multiplicando (38) por u, obtemos

$$u(f(\beta_2) - f(\beta_1)) \equiv (\beta_2 - \beta_1) \pmod{p^k},$$

e o resultado segue de (ii).

2. O algoritmo

Vamos aplicar os resultados da seção anterior a um algoritmo que detecta se um dado inteiro positivo a tem raiz n-ésima inteira. O algoritmo pressupõe que tanto a, quanto n, sejam inteiros ímpares. Por isso, analisaremos antes o que fazer se são pares.

Se a for par, podemos extrair a maior potência de 2 que divide a e depois, se necessário, aplicar o algoritmo ao co-fator. Isto é feito dividindo a sucessivamente por 2 até obter um quociente ímpar. Para isso teremos que executar, no máximo, $\lfloor \log_2 a \rfloor$ divisões, ao custo de $O((\log_2 a)^2)$ cada uma; o que dá um custo total de $O((\log_2 a)^3)$. Observe que, se a multiplicidade de 2 na fatoração de a não for divisível por n, então a não pode ter raiz n-ésima inteira e não precisamos sequer aplicar o algoritmo.

Por outro lado, se n for par, então precisamos extrair algumas raízes quadradas antes de poder aplicar o algoritmo. O número de raízes quadradas que precisamos extrair é igual à multiplicidade de 2 na fatoração de n. Podemos

modificar facilmente o algoritmo descrito abaixo de modo a extrair raízes quadradas. Para mais detalhes veja o exercício 3.

Resta-nos, portanto, descrever um algoritmo polinomial, para o caso em que a e n sejam ímpares. O algoritmo utiliza o método de Newton da seção anterior para o primo 2. Com isto estaremos operando módulo potências de 2, o que explica porque a não pode ser par.

Algoritmo para detectar potências n-ésimas

Entrada: inteiros positivos ímpares a e n.

Saída: sim ou não, dependendo se a tem ou não raiz n-ésima inteira.

Etapa 1: Seja $k = \lceil (\log_2 a)/n \rceil$ e faça $r = \lceil \log_2 k \rceil$.

Etapa 2: Inicialize $S = 1$ e $i = 0$.

Etapa 3: Se $i = r$ páre de executar o laço e calcule $R = S^n - a$. Imprima *sim*, se $R = 0$, ou *não*, se $R \neq 0$.

Etapa 4: Calcule o inverso T de nS^{n-1} módulo 2^{2^i} pelo algoritmo euclidiano estendido.

Etapa 5: Atribua a S o resíduo módulo 2^{2^i} de $S - (S^n - a)T$ e volte à etapa 3.

Vamos, primeiramente, demonstrar que o algoritmo funciona corretamente. Seja S_j o valor da variável S no j-ésimo laço da execução do algoritmo. Sendo $f(x) = x^n - a$, chamaremos de $\mathbf{A}(j)$ ao seguinte conjunto de três afirmações sobre S_j:

- $S_j \equiv 1 \pmod{2^{2^j}}$,
- $f(S_j) \equiv 0 \pmod{2^{2^j}}$, e
- $f'(S_j)$ é inversível módulo p.

Pelo teorema A.3 se $\mathbf{A}(j)$ for verdadeira, então $\mathbf{A}(j+1)$ também será. Mas a variável S é inicializada com $S_0 = 1$ e n e a são ímpares, de modo que

$$f(1) \equiv 0 \pmod 2 \text{ e } f'(1) \equiv n \equiv 1 \pmod 2.$$

Portanto, $\mathbf{A}(0)$ é verdadeira. Logo, por indução em j, $\mathbf{A}(j)$ é verdadeira, para todo $j \geq 0$. Em particular, depois do algoritmo executar r laços teremos que

$$S_r^n \equiv a \pmod{2^{2^r}}.$$

Mas, se $a = \alpha^n$ para algum inteiro $\alpha \in \mathbb{Z}$, isto implica que

$$S_r^n \equiv \alpha^n \pmod{2^{2^r}};$$

e, pelo teorema A.4, concluímos que

$$S_r \equiv \alpha \pmod{2^{2^r}}.$$

Entretanto,

$$\alpha = a^{1/n} < 2^k < 2^{2^r}.$$

Mas dois inteiros não negativos menores que 2^{2^r} só podem ser congruentes módulo 2^{2^r} se forem iguais. Portanto, a é a n-ésima potência de α se, e somente se $S_r = \alpha$, provando que o algoritmo funciona corretamente.

Falta determinar o custo do algoritmo. Começaremos analisando cada uma de suas etapas individualmente. Como os números estão armazenados em notação binária, o custo da etapa 1 é desprezível, a não ser pela divisão por n que tem custo $O((\log_2 n)^2)$.

O custo das etapas 3, 4 e 5 depende diretamente do custo de calcular uma potência n-ésima módulo 2^{2^r}. Mas, pela seção 3 do capíulo 4, isto pode ser feito ao custo

$$O(2^{4r} \log_2 n) = O(k^4 \log_2 n) = O((\log_2 a)^4 \log_2 n).$$

Na etapa 3 são executadas duas operações, uma potenciação com expoente n, e uma soma. A soma tem um custo muito mais baixo que o da potenciação, e pode ser ignorada. Como $S_r^n < 2^{2^r}$, então S_r^n não pode ter custo maior que o de elevar uma classe módulo 2^{2^r} à n-ésima potência. Mas, já vimos que isto custa $O((\log_2 a)^4 \log_2 n)$.

Como $i \leq r$, podemos analisar as duas etapas que faltam supondo que $i = r - 1$. As operações de adição e multiplicação módulo 2^{2^r} têm custo inferior a

$$O((\log_2(2^{2^r})^2) = O((\log_2 a)^2).$$

Já o cálculo do inverso pelo algoritmo euclidiano estendido, tem um custo $O((\log_2(a)^3)$, segundo o exercício 7 do capítulo 1. Portanto, podemos ignorá-las, quando comparadas à exponenciação. Logo, o custo das etapas 4 e 5 é o mesmo que o do cálculo da potência que, mais uma vez, é da ordem de $O((\log_2 a)^4 \log_2 n)$.

Não podemos esquecer que as duas últimas etapas são executadas r vezes. Como r é menor que $1 + \log_2 \log_2 a$, o custo total do algoritmo não pode execeder

$$O((1 + \log_2 \log_2 a)((\log_2 a)^4 \log_2 n)) = O((\log_2 a)^6),$$

uma vez que $n \leq \log_2 a$. Portanto, não há dúvida de que se trata de um algoritmo polinomial.

3. Exercícios

1. Use o algoritmo da seção 2 para calcular a raiz quinta de 28153056843.

2. Modifique o algoritmo descrito na seção 2 para que faça todas as reduções módulo 3 e mostre como usá-lo para calcular raízes quadradas.

3. Use o algoritmo do exercício 2 para achar a raiz quadrada de 15733939225.

4. Mostre que o algoritmo descrito no exercício 2 tem custo polinomial.

Bibliografia

Agrawal, M., N. Kayal e N. Saxena
2002 *Primes is in P*, IIT, preprint, disponível em
`http://www.cse.utk.ac.in/ news/primality.pdf`

Alford, Granville e Pomerance
1994 *There are infinitely many Carmichael numbers*, Ann. of Math. **140**, 703–722.

Bernstein, D. J.
1998 *Detecting Perfect Powers in Essentially Linear Time*, Math. of Computation **67** (223), 1253–1283.

2003a *Proving Primality after Agrawal-Kayal-Saxena*, versão 20030125 do preprint, disponível em `http://cr.yp.to/djb.html`.

2003b *proving primality in essentially quartic random time*, preprint, versão 20030417, disponível em `http://cr.yp.to/djb.html`.

Berrizbeitia, P.
2002 *Sharpening 'Primes is in P' for a large family of numbers*, preprint disponível em arXiv:math.NT/0211334.

Blake, I., G. Seoussi e N. Smart
2000 *Elliptic curves in cryptography*, Cambridge University Press, Cambridge.

Bornemann, F.
2003 *Primes Is in P: Breakthrough for "Everyman"*, Notices of the Amer. Math. Soc. **50** (5), 545–552.

J. S. Cohen
2003 *Computer algebra and symbolic computation: mathematical methods*, AK Peters, Natick.

Cormen, T. H., C. E. Leiserson e R. L. Rivest
1999 *Introduction to Algorithms*, The MIT Press, Cambridge, Massachusetts.

Coutinho, S. C.
2001 *Números inteiros e Criptografia RSA*, Série Computação e Matemática, IMPA, Rio de Janeiro.

Crandall, R. e J. Papadopoulos
2003 *On the implementation of AKS-class primality tests*, preprint, disponível em
`http://developer.apple.com/hardware/ve/pdf/aks3.pdf`.

Crandall, R. e C. Pomerance
2002 *Prime Numbers: a computational perspective*, Springer, New York.

Gonçalves, A.
1979 *Introdução à álgebra*, Projeto Euclides, IMPA, Rio de Janeiro.

Hardy, G.H. e E.M. Wright
1994 *An introduction to the theory of numbers*, quinta edição, Oxford Science Publications, Oxford University Press, Oxford.

Ingham, A. E.
1932 *The distribution of primes*, Cambridge University Press, Cambridge.

D. E. Knuth
1981 *The art of computer programming: seminumerical algorithms*, volume 2, segunda edição, Addison-Wesley, Reading.

Landau, E.
2002 *Teoria Elementar dos Números*, Coleção Clássicos da Matemática, Editora Ciência Moderna, Rio de Janeiro.

J. H. McKay
1959 *Another proof of Cauchy's group theorem*, Amer. Math. Monthly **66**, 119.

P. Mihăilescu e R. M. Avanzi
2003 *Efficient "quasi"-deterministic primality test improving AKS*, preprint disponível em
`http://www-math.uni-paderborn.de/~preda/`.

Monier, L.
1980 *Evaluation and comparison of two efficient probabilistic primality testing algorithms*, Theoret. Comp. Sci. **12**, 97–108.

Morain, F.
2002 *Primalité Théorique et Primalité Pratique, ou AKS vs ECPP*, preprint disponível em `http://www.lix.polytechnique.fr/Labo/Francois.Morain`.

2003 *La Primalité en temps polynomial*, a ser publicado no Séminaire Bourbaki 2002/2003, preprint disponível em `http://www.lix.polytechnique.fr/Labo/Francois.Morain`.

Nair, M.

1982 *On Chebyshev-type Inequalities for Primes*, Amer. Math. Monthly **89**, 126–129.

Rabin, M.
1980 *Probablistic algortihm for testing primality*, J. Number Theory **12**, 128–138.

Ribemboim, P.
2001 *Números primos: mistérios e recordes*, Coleção Matemática niversitária, IMPA, Rio de Janeiro.

J. Ritter
1997 *Chacun sa vérité: les mathématiques en Egypte et en Mésopotamie*, em Élements d'Histoire des Sciences, editado por M. Serres, Larousse, Paris.

Schoof, R.
2003 *Agrawal-Kayal-Saxena primality test*, disponível em
`http://www.mat.uniroma2.it/~schoof/agrawalma.pdf`.

Silverman, J. H. e J. Tate
1992 *Rational points on elliptic curves*, Springer-Verlag, New York.

Szwarcfiter, J. L.
1986 *Grafos e algoritmos computacionais*, Editora Campus, Rio de Janeiro.

Voloch, J. F.
1987 A distribuição dos números primos, *Matemática Universitária*, número 06, 71–82.

von zur Gathen, . e J. Gerhard
1999 *Modern Computer Algebra*, Cambridge University Press, Cambridge.

Índice

Abel, N. H., 4, 26
Adição, 41
  de classes, 14
  de inteiros, 12
  de polinômios, 43
  em uma curva elíptica, 90
Agrawal, M., 7
Alexandria, 1
Algarismo binário, 12
Algoritmo
  AKS, 7, 18, 57, 62, 70, 75, 80, 89
  ciclotômico, 7
  de exponenciação, 84
  de fatoração, 11, 15, 90
    de polinômios, 57
  euclidiano, 22
    estendido, 55, 96, 97
  exponencial, 15, 65
  polinomial, 15
Algoritmos
  probabilísticos, 91
Anel, 7, 8, 41
  comutativo, 42
  de polinômios, 43, 51, 61
  quociente, 8, 57
Aristóteles, 1
Artjuhov, M., 6, 67
Associatividade, 26, 37
associatividade, 28

Base, 65
Binômio de Newton, 62
Bit, 12, 65

Chebyshev, P. L., 3, 7, 21
Coeficiente, 43
  líder, 43
Complexidade, 7
Composto, 1, 9
Comutatividade, 26, 37, 42
comutatividade, 29
Congruência
  módulo n, 34
  módulo um subgrupo, 34
Corpo, 7, 8, 41, 42, 50, 51, 54, 57, 58
  finito, 53, 54, 57
Crandall, R., 90
Criptografia, 6
  e curvas elípticas, 90
Crivo de Eratóstenes, 2, 8, 9, 16, 21, 23, 81, 83
  custo, 17
Curva
  elíptica, 89
  plana, 89
curva
  não singular, 89
Custo, 12, 70
  da adição de polinômios, 44
  do algoritmo, 97
  do algoritmo AKS, 82
  polinomial, 57, 93

de La Vallé Poussin, C. J., 3
Dedekind, R., 3
Diofanto, 2, 4
Disquisitiones Arithmeticae, 2, 3
Divisão, 14
  com resto, 45
  de polinômios, 45
Divisor, 46, 59
  próprio, 59
Domínio, 42

Egípcios, 68
El Gamal, 6
Elemento
  inverso, 26, 42
  invertível, 59
  irredutível, 59

neutro, 26, 37, 42
simétrico, 26, 42
Elementos, 1, 2
ENIAC, 5
Estrutura algébrica, 25
Euclides, 1, 2
Euler, L., 2, 21
Exponenciação, 97

Fórmula
de Taylor, 93
Física, 26
Fator, 46
Fermat, P., 2, 3, 6
Fermat, S., 4
FFT, 90
Função π, 3

Galois, E., 4
Gauss, C. F., 1–3, 10
Gerador, 30, 31, 50
Grécia, 1, 2, 10
Grau, 43
Grupo, 5, 7, 25
abeliano, 25, 26, 29, 38, 42, 90
aditivo, 26, 51
cíclico, 31, 32, 38, 50
de ordem prima, 33
finito, 29
multiplicativo, 26, 42
quociente, 8, 37, 41, 49, 51, 77

Hadamard, J., 3
Hartmanis, J., 7

Ideal, 4, 8, 49, 58
gerado por um conjunto, 50
principal, 50, 59
Igualdade
de polinômios, 43
Indeterminada, 43
Inteiros módulo n, 42

Kakapo, 25
Kayal, N., 7
Kummer, E. E., 3

Lagrange, J. L., 4
Lamé, G., 22
Legendre, 3
Lehmer, D. H., 5, 89
Leibniz, G. W., 65
Lema
chave, 33
Lenstra Jr, H. W., 7
Lenstra, H. W., 80
Lucas, E., 71, 89
Lucas, F. E. A., 5

Máximo divisor comum, 22, 48
Método
de Newton, 81, 93, 94, 96
Memória, 13, 18
Miller, G., 6
Miller, G. L., 67
Multiplicação, 41
de classes, 14, 36
de inteiros, 13
de polinômios, 44
por segmentação binária, 90

Número
binomial, 18, 23, 61
complexo, 26, 42
de Carmichael, 65, 67
de Fermat, 72
de Mersenne, 5
inteiro, 26, 42
racional, 26, 42
real, 26, 42
Newton, I., 8
Nicômaco, 2
Notação O, 14

Operação, 25, 41
aritmética elementar, 12
com classes, 14
no grupo quociente, 36
Operações
aritméticas, 64
Ordem
de um elemento, 30
de um grupo, 27

Papadopoulos, J., 90
Peneira, 2
Pentatlos, 2
Polinômio, 41
irredutível, 54
mônico, 43, 52
redutível, 54
Polinômios, 8, 42
Potência, 29
Primes is in P, 7
Primo, 1, 9, 42, 54, 66
Primorial, 20
Primos
infinidade, 22
Princípio
da Casa do Pombo, 30

Pseudoprimo, 72
  forte, 72

Quociente, 45

Récréations Mathématiques, 5
Rabin, M., 6, 8, 67
Raiz, 48
  n-ésima, 95
Raiz quadrada, 11
Relação
  de equivalência, 34
  reflexiva, 34
  simétrica, 34
  transitiva, 34
Reserva, 12
Resto, 45
RSA, 6

Saxena, N., 7
Selfridge, J., 6, 67
Seqüência
  de Fibonacci, 22
Stearns, R. E., 7
Subgrupo, 28, 41

Tabela de multiplicação, 27
Tempo de de execução, 12
Teorema
  Último de Fermat, 3, 90
  da Fatoração Única, 1, 2, 7–10, 21, 54
  de Cauchy, 39
  de Fermat, 63, 65, 71, 72
  de Lagrange, 31, 37, 77
  de Wilson, 73
  dos Números Primos, 3
Teste
  da curva elíptica, 89, 90
  de primalidade, 61, 64
    determinístico, 64
    probabilístico, 68
  forte, 67
  forte de composição, 66
  não determinístico, 64
Transformada
  rápida de Fourier, 90

Variável, 43

Wiles, A., 4, 90
Wilson, J., 7